UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL

ENHANCING iDynoMiCS FRAMEWORK WITH A PLASMID CONJUGATION MODULE

AUTOR: AURORA SOFÍA ARROYO GARCÍA

TUTOR: ALFONSO RODRÍGUEZ-PATÓN ARADAS

MASTER FINAL PROJECT

July 2013

Table of Contents

Figure Index

Resumen

Actualmente existen aplicaciones que permiten simular el comportamiento de bacterias en distintos hábitats y los procesos que ocurren en estos para facilitar su estudio y experimentación sin la necesidad de un laboratorio.

Una de las aplicaciones de software libre para la simulación de poblaciones bacteriológicas mas usada es iDynoMiCS (individual-based Dynamics of Microbial Communities Simulator), un simulador basado en agentes que permite trabajar con varios modelos computacionales de bacterias en 2D y 3D. Este simulador permite una gran libertad al configurar una numerosa cantidad de variables con respecto al entorno, reacciones químicas y otros detalles importantes. Una característica importante es el poder simular de manera sencilla la conjugación de plásmidos entre bacterias.

Los plásmidos son moléculas de ADN diferentes del cromosoma celular, generalmente circularles, que se replican, transcriben y conjugan independientemente del ADN cromosómico. Estas están presentes normalmente en bacterias procariotas, y en algunas ocasiones en eucariotas, sin embargo, en este tipo de células son llamados episomas.

Dado el complejo comportamiento de los plásmidos y la gama de posibilidades que estos presentan como mecanismos externos al funcionamiento básico de la célula, en la mayoría de los casos confiriéndole distintas ventajas evolutivas, como por ejemplo: resistencia antibiótica, entre otros, resulta importante su estudio y subsecuente manipulación.

Sin embargo, el marco operativo del iDynoMiCS, en cuanto a simulación de plásmidos se refiere, es demasiado sencillo y no permite realizar operaciones más complejas que el análisis de la propagación de un plásmido en la comunidad. El presente trabajo surge para resolver esta deficiencia de iDynomics. Aquí se analizarán, desarrollarán e implementarán las modificaciones necesarias para que iDynomics pueda simular satisfactoriamente y mas apegado a la realidad la conjugación de plásmidos y permita así mismo resolver distintas operaciones lógicas, como lo son los circuitos genéticos, basadas en plásmidos.

También se analizarán los resultados obtenidos de acuerdo a distintos estudios relevantes y a la comparación de los resultados obtenidos con el código original de iDynomics. Adicionalmente se analizará un estudio comparando la eficiencia de detección de una sustancia mediante dos circuitos genéticos distintos.

Asimismo el presente trabajo puede tener interés para el grupo LIA de la Facultad de Informática de la Universidad Politécnica de Madrid, el cual está participando en el proyecto europeo BACTOCOM que se centra en el estudio de la conjugación de plásmidos y circuitos genéticos.

Abstract

Currently there are applications that simulate the behavior of bacteria in different habitats and the ongoing processes inside them to facilitate their study and experimentation without the need for an actual laboratory.

One of the most used open source applications to simulate bacterial populations is iDynoMiCS (individual-based Dynamics of Microbial Communities Simulator), an agent-based simulator that allows working with several computer models of 2D and 3D bacteria in biofilms. This simulator allows great freedom by means of a large number of configurable variables regarding environment, chemical reactions and other important details of the simulation. Within these characteristics there exists a very basic framework to simulate plasmid conjugation.

Plasmids are DNA molecules physically different from the cell's chromosome, commonly found as small circular, double-stranded DNA molecules that are replicated, conjugated and transcribed independently of chromosomal DNA. These bacteria are normally present in prokaryotes and sometimes in eukaryotes, which in this case these cells are called episomes.

Plasmids are external mechanisms to the cells basic operations, and as such, in the majority of the cases, confer to the host cell various evolutionary advantages, like antibiotic resistance for example. It is imperative to further study plasmids and the possibilities they present.

However, the operational framework of the iDynoMiCS plasmid simulation is too simple, and does not allow more complex operations that the analysis of the spread of a plasmid in the community. This project was conceived to resolve this particular deficiency in iDynomics, moreover, in this paper is discussed, developed and implemented the necessary changes to iDynomics simulation software so it can satisfactorily and realistically simulate plasmid conjugation, and allow the possibility to solve various logic operations, such as plasmid-based genetic circuits.

Moreover the results obtained will be analyzed and compared with other relevant studies and with those obtained with the original iDynomics code. Conjointly, an additional study detailing the sensing of a substance with two different genetic circuits will be presented.

This work may also be relevant to the LIA group of the Faculty of Informatics of the Polytechnic University of Madrid, which is participating in the European project BACTOCOM that focuses on the study of the of plasmid conjugation and genetic circuits.

Part 1 Introduction

Chapter 1 Introduction

The present project will describe the modifications and extensions of functionality done to the bacterial simulator iDynomics to make it able to reproduce plasmid conjugation in bacteria during biofilm formation, as well as a short experiment as a proof of concept.

Overview

Simulation software is based on the process of imitating a real phenomenon with a set of mathematical formulas; this confers a certain degree of accuracy and allows the user to observe a set of operations without actually performing them.

The iDynoMiCS software is one of the most known open source bacterial simulators available; it simulates the growth of microbial communities, easily allowing the user to specify many different types of simulations. It also has a small module to simulate plasmid conjugation; however, it lacks the functionality to reproduce complex plasmid operations.

Plasmids are extra-chromosomal genetic elements found in almost all bacterial species, which among other conjugative elements are the key vectors of horizontal gene transfer and essential tools in genetic engineering, they often code genes involved in a plethora of additional functions conferred to the bacterial hosts. [1]

To solve this particular iDynoMiCS restraint, the present project will describe modifications made to the software in order for it to simulate more complex plasmid conjugations.

As iDynomics is an open source software, all modifications made in this project will be made publicly available.

Furthermore, a study regarding the detection of a substance with two distinct genetic circuits using this simulator will be presented.

Objectives

The aim of this project is to enhance the iDynomics software with a robust framework to simulate more complex plasmid conjugation and operations. In order to satisfy this objective four goals were defined:

- Design and implement a model for multiple plasmid support in a single bacteria
- Design and implement growth, division, and death processes of bacterial hosts
- Design and implement a model for proteins and other substances that incur in the conjugation process.
- Enhance the current conjugation algorithm to reflect desired behavior.
- The resultant model is successfully validated as correct.

Thesis Plan

Plasmids and Horizontal Gene Transfer (HGT) are explained in Chapter 3. During Chapter 4, the current iDynoMiCS class structure, processes and simulation flow will be explained. A detailed view of the problem will be described in Chapter 5, and the new model detailing the processes, classes and simulation flow in Chapter 6. In Chapter 7 the results of simulations are presented; first the results of the validation performed, afterwards the analysis and results of the experiment on the sensibility of a classic AND plasmid sensor vs. mobile AND will be presented. Finally, in Chapter 8 the conclusions and future work are exposed.

Chapter 2 Problem description

iDynoMiCS is a framework that offers the possibilities to successfully simulate bacterial colony growth and interaction between individual agents. As described in the previous chapter, iDynomics has the functionality to simulate bacterial donors for plasmids as well as plasmid conjugation, though it is by far a very simple implementation.

One of the things iDynoMiCS is deficient is the correct representation of the plasmid in a more real-like modular way, especially pertaining to the mobility and conjugation mechanisms. Also, it lacks the ability to code and express genes that would be part of the operative module of the plasmid, sometimes required to inhibit or activate other important functions. This shortcoming makes iDynoMiCS expressly incapable of simulating plasmid-based genetic circuits.

To solve this plasmid model insufficiency, the present work will start analyzing state-of-the-art papers about plasmid conjugation process, and then implement the required changes into the iDynomics simulation flow. Fortunately iDynomics was designed with a modular mindset, making the changes required to be done in a manner that ensures that core functionality is not affected.

Part 2 Theoretical framework

Chapter 3 Plasmids and Horizontal Gene Transfer

Plasmids

Plasmids are extra-chromosomal genetic elements found in almost all bacterial species, which among other conjugative elements are the key vectors of horizontal gene transfer and essential tools in genetic engineering. Among the phenotypes conferred by different plasmids are: resistance to and/or production of antibiotics, production of several substances such as: bacteriocins (proteins that can kill cells of the same or closely related species), enterotoxins, virulent factors; also the degradation of complex organic compounds, detoxication, ecological interactions, etc. Hence, an understanding of plasmid mobility is essential to an understanding of the evolution of these important bacterial traits, often involved in human health or well-being (Lipps, 2008, Bailey, et al. 2005).

Plasmids are mostly found on prokaryotic cells; meanwhile, plasmids in eukaryotic bacteria are called Episome and are replicated inside the nucleus but remain physically separate from host cell chromosomes (Lipps, 2008).

The most common structure of plasmids is that of extra-chromosomal covalently closed circular (and supercoiled) units of DNA, although several examples of linear plasmids also exist (Norman, 2009). In fact, the most important defining feature of plasmids is their separateness from the host chromosome and hence their ability to replicate autonomously. This ability provides autonomy from host replication and also offers an abundance of alternative hosts and evades the danger of becoming extinct with one particular cell population if environmental conditions were to suddenly change unfavorably (Norman, 2009).

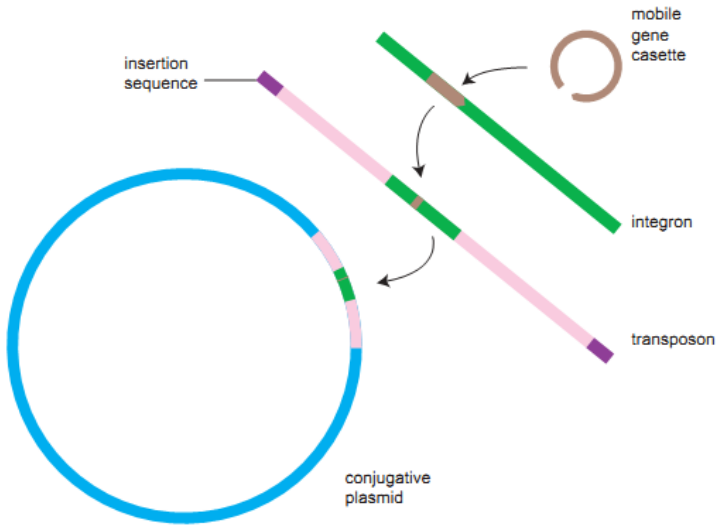


FIGURE 1-THE MODULAR AND HIERARCHICAL COMPOSITION OF MGES. A. NORMAN ET AL. REVIEW. CONJUGATIVE PLASMIDS

Plasmids, along with other mobile genetic elements (MGEs) are components of the horizontal gene pool (such as insertion sequences, transposons, integrons, bacteriophages, genomic islands and combinations of these elements)and can be exchanged promiscuously between a broad spectrum of bacteria and contribute to bacterial genome flexibility. Plasmids are also molecular biology workhorses whose mobility opened up the possibility of genetic manipulation (Norman, 2009).

More and more evidence is emerging in support of significant gene shuffling by horizontal gene transfer (HGT), since these elements can be found among members of the three major domains: *Archaea*, *Bacteria*, and *Eukarya* — including plants, fungi and human cells. Despite plasmids presence almost everywhere only a limited amount of plasmid-sequence data is currently available. Comparative analyses have revealed the complexity of plasmid genetics, the capacity of these elements to replicate autonomously, the presence of a unique set of genes that are clearly distinct from the genes that are typically found on bacterial chromosomes (Bailey, et al. 2005).

The relationship between plasmids and their hosts is crucial for host ecology, since plasmids allow bacterial populations to sample the horizontal gene pool for adaptive traits that might be advantageous for survival under local selective pressures, they also provide genetic variation, act as sources for recombination and, owing to their longer retention time in the cell, can allow faster gene fixation than either transformation or transduction, leading to a greater likelihood that the new trait will persist (Norman, 2009, Bailey, et al. 2005).

Horizontal Gene Transfer

To facilitate the study of microbial genetics, Norman et al. (2010) proposed the term supergenome, which refers to the total pool of genes available to a prokaryotic organism within a particular setting, or the horizontal gene pool, it contains both the private pool, that consists of the fixed and distinctive genes encoded on the chromosome of the prokaryote; and the communal pool comprehended by genes encoded on MGEs that are available to all prokaryotes in the community (figure 2). Genes located on MGEs and encoding traits giving periodic selective advances could be recruited and lost by individual cell lines multiple times (Norman, 2009).

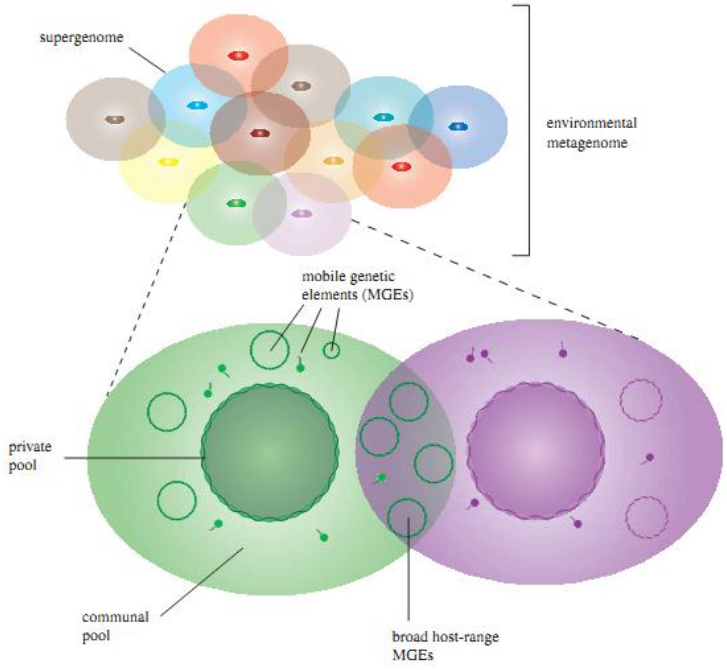


FIGURE 2 -THE SUPERGENOME IS THE TOTAL POOL OF GENES READILY AVAILABLE TO AN ORGANISM IN A PARTICULAR COMMUNITY SETTING, ALSO KNOWN AS THE HORIZONTAL GENE POOL . IT CONSISTS OF GENES IN THE PRIVATE POOL, THAT IS, ESSENTIAL GENES ENCODED ON THE CHROMOSOME, AND THE COMMUNAL POOL, WHICH CONSISTS OF GENES ENCODED ON MGES (PLASMIDS, TRANSPOSONS, VIRUSES ETC.). A. NORMAN ET AL. REVIEW. CONJUGATIVE PLASMIDS

Horizontal Gene Transfer (HGT) is the transfer of genetic material, generally MGEs in the communal pool, from one cell to another. HGT is a major force impacting the adaptative evolution and rapid adaptation of bacteria, and may occur by three different mechanisms: Transformation, transduction and conjugation. the latter is thought to be quantitatively more important (Norman, 2009).

HGT by transformation refers to the uptake and expression of foreign genetic material (DNA or RNA) into cells from the surrounding environment taken up through the cell membrane(s) and relies on the presence of plasmid or chromosomal DNA fragments that are often released as a result of cell death or active excretion. This process is relatively common in bacteria, but less so in eukaryotes, this is the only prokaryotic HGT process that relies uniquely in the physiological status of the host (Bailey, et al. 2005, Seoane, 2010).

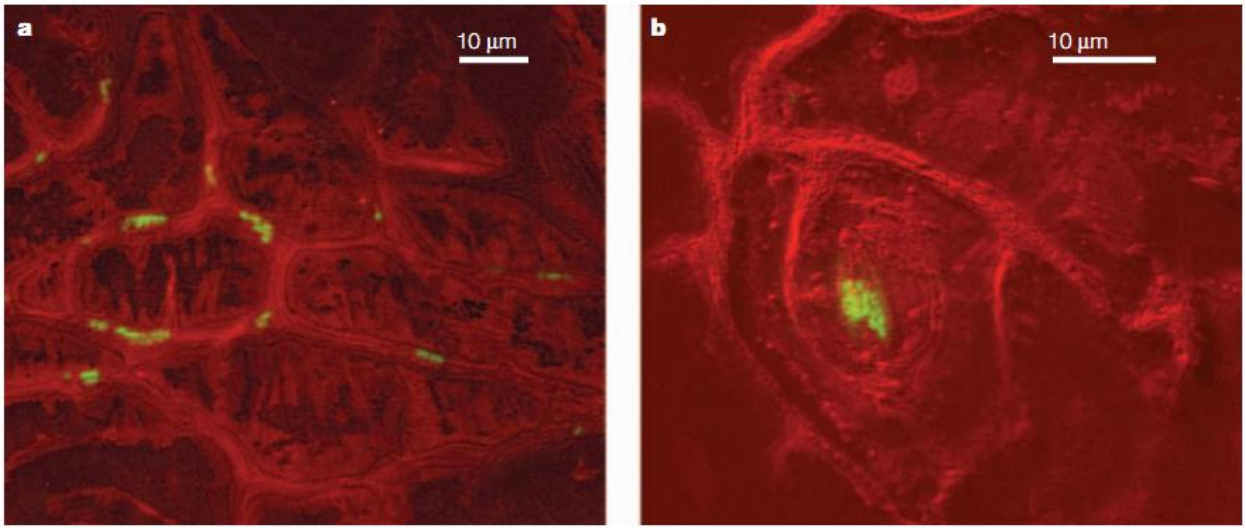


FIGURE 3- EXAMPLE OF HORIZONTAL GENE TRANSFER (HGT), CONFOCAL SCANNING LASER MICROGRAPHS SHOWING GREEN FLUORESCENT TRANSCONJUGANT CELLS IN THE INTERSTICES OF EPIDERMAL CELLS (A) AND INSIDE A STOMA (B) OF A BEAN LEAF. S. J. SØRENSEN ET AL, STUDYING PLASMID HORIZONTAL TRANSFER IN SITU: A CRITICAL REVIEW

Transformation is often used in laboratories to insert novel genes into bacteria for experiments or for industrial or medical applications. Relatively little is known about the predominance and evolutionary history of taxonomic distribution of organisms possessing this property. Thus, whether natural transformation only benefits a limited number of species or has a large impact on gene flow in nature remains unknown (Bailey, et al. 2005, Seoane, 2010).

Transduction is the process by which DNA is transferred from one bacterium to another by a bacteriophage. This phenomenon generally occurs from errors in the packaging of DNA and can occur in the natural environment where phages are abundant and diverse. The transduction mechanism requires that a phage replicates within the donor organism and, in the process of DNA packaging, occasionally incorporates DNA fragments from the host into the phage's protein shell. Phages are then released into the environment in which they can inject their DNA into a new host (Bailey, et al. 2005, Norman, 2009).

Transduction does not require physical contact between the cell donating the DNA and the cell receiving the DNA and is a common tool used by molecular biologists to stably introduce a foreign gene into a host cell's genome (Smillie, et al., 2010).

Conjugation is a mechanism that directly transfers DNA between bacterial cells during cell-to-cell contact; therefore, it would seem like a safe way to transport large DNA fragments with high fidelity between physically proximate donors and recipients. Some environments are believed to be more conductive for conjugation than others, like microbial biofilms with high density of highly active bacterial cells. Likewise, the products of conjugational genes seem to promote cell-to-cell contact, which can facilitate biofilm formation (Bailey, et al. 2005, Smillie, et al., 2010).

It is one of the main forces contributing to shape bacterial genomes and the main goal to simulate in this project. Perhaps more importantly, it is responsible for the rapid spread of antibiotic resistance (AbR) among bacterial pathogens since some plasmids can conjugate between remotely related organisms belonging even to different kingdoms (Lipps,2008).

As a means of ensuring genetic variation in bacteria, the recruitment of new genes by HGT could therefore be seen as form of compensating for the lack of sexual recombination, which is why it is often referred to as “bacterial sex” (Norman, 2009).

Biofilms

A Biofilm is a community of microorganisms attached to a surface, surface in which cells adhere to one another. These attached cells are frequently embedded with a matrix of self-produced extracellular polymeric substance (EPS), various organic and inorganic particles, and water. Biofilms exist at solid–liquid, solid–gas, liquid–liquid or liquid–gas interfaces (Bailey, et al. 2005).

Bacterial biofilms can sustain a high density of bacteria, protect against any harmful substances (like antimicrobials) or other fluctuations in the environment, bacteriophage attacks and also provide a frame for HGT, since they provide a stable habitat for direct contact between cells and extracellular DNA contributes to the maintenance of the structure. Biofilm formation might even represent the default state of most bacteria (Bailey, et al. 2005, Smillie, et al., 2010).

Conjugation

Plasmids have a dominant role in the horizontal gene transfer between bacteria; they can transfer DNA between genera, phyla and major domains by a mechanism called conjugation. Conjugation basically involves direct cell-to-cell contact, a mating-pair formation and DNA exchange mediated by special filaments. Conjugation can also prevent plasmid loss by reinfection plasmid-free cells (Bailey, et al. 2005).

In more detail, conjugation could be better analyzed if separated in two functions: mating pair formation (MPF) and relaxosome formation (or MOB genes). MPF is the first requirement for conjugation; it consists in the donor and recipient cells connecting physically by conjugation filaments (pilli). In most cases, this occurs through the synthesis of a type IV secretion system (T4SS), which generates the pili that extends from the donor cell to suitable recipient cells within its proximity (Norman, 2009).

The assembly of a T4SS often requires expression of 10 or more genes that are usually encoded on a single operon, in addition to conjugation, T4SS are also involved in DNA uptake and release in bacteria and also in the translocation of proteins into host cells during infection. Usually, the expression of T4SS modules is repressed. Its expression can normally be triggered by both external cues (like a change in temperature for example) and encountering plasmid-free cells, which results in the epidemic spread of the plasmid through whole plasmid-free populations (Bailey, et al. 2005).

T4SS' pilus can be thick flexible, thin flexible or rigid, previously served as a way of distinguishing plasmid incompatibility groups (Norman, 2009). This filament tends to delimit the extent of gene dissemination within the horizontal gene pool because it determines the conjugation host range, which cells a donor can physically connect with, and the preferred medium of transfer. It is arguable that the presence of conjugative pili, and no other conjugative functions or HGT itself, provides a direct selective advantage to the host cell (Bailey, et al. 2005).

The second function in the conjugation process involves several steps; first a protein called relaxasa, that is considered the key protein in conjugation, mediates the nicking of the plasmid origin of transfer (oriT), then the relaxosome is formed. A relaxosome is the single strand of the plasmid DNA that will be transferred. Then, a related coupling protein helps the relaxosome dock with T4SS, where it is transported to the recipient cell. Finally, the second strand of the plasmid is reconstituted in both the donor and trans-conjugant (Seoane, 2010, Smillie, et al., 2010).

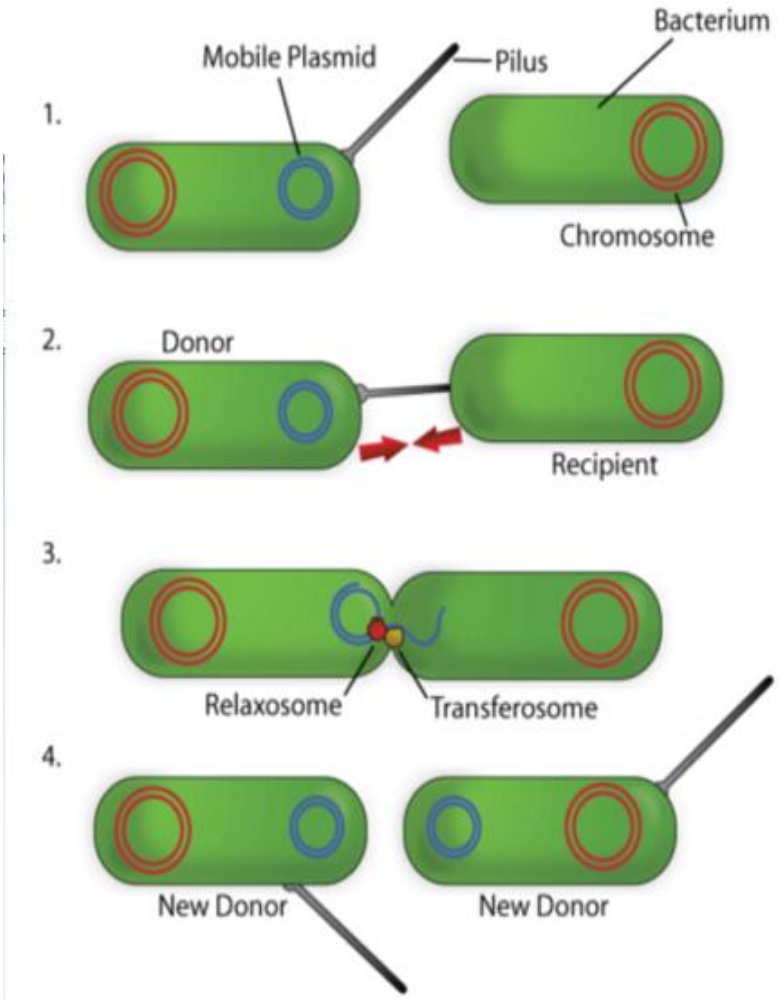


FIGURE 4- CONJUGATIVE TRANSFER AT THE INDIVIDUAL CELL LEVEL. THE BEGINNING OF A CONJUGATIVE EVENT IS GIVEN BY THE RECOGNITION AND BINDING OF THE RECIPIENT CELL SURFACE BY THE DONOR PILUS (1). AFTER BINDING, THE PILUS RETRACTS (2) AND MATING PAIR STABILIZATION (MPS) RESULTS IN A STABLE ASSOCIATION BETWEEN DONOR AND RECIPIENT CELLS (3) WHICH FAVORS THE SUCCESSFUL COMPLETION OF THE CONJUGATION PROCESS (4). J. SEOANE, "INDIVIDUAL-BASED ANALYSIS AND PREDICTION OF THE FATE OF PLASMIDS IN SPATIALLY STRUCTURED BACTERIAL POPULATIONS: PHD THESIS,"

Surface and entry exclusion.

Entry exclusion is the property by which some plasmids physically prevent the entry of similar plasmids into the host cell. This is usually achieved by either surface exclusion or entry exclusion. Surface exclusion works by interfering with the attachment of the conjugation pilus to the surface of the recipient cell, while entry exclusion prevents the transfer of DNA between existing mating pairs. These methods of exclusion function as a safeguard against incompatibility; moreover, they can benefit the host cell by avoiding further metabolic burden. In most cases, the genes encoding entry exclusion are parts of the MPF modules since they are embedded within the T4SS operons (Seoane, 2010, Smillie, et al., 2010, Murphy, 2012).

Plasmid Transfer Efficiency

To further analyze the reach of plasmids and other MGE's that operate in bacterial populations is required a precise quantitative determination of conjugation rates. This data will provide essential parameters for the development of mathematical models to explain plasmid dissemination and the effects of conjugation. In addition, the large variety of different metrics still used to report plasmid-transfer frequencies strongly remarks the need of a standardized metric for plasmid-transfer efficiency (Del Campo, et al., 2012).

The most widely used indicator is the transfer frequency, that is, the ratio of the number of trans-conjugants to either the number of donors or of recipients, although ideally, it should be reported as the number of transfer events per donor–recipient encounters, since such a ratio would allow true comparisons of the transfer efficiencies of different plasmids or in different environmental settings (Smillie, et al., 2010, Del Campo, et al., 2012).

Conjugative Plasmids

Mobility is a crucial part of plasmid fitness. The two functions that are deemed essential for plasmid survival are DNA replication and horizontal spread. A set of mobility (MOB) genes is a necessary requirement as it allows conjugative DNA processing. In conjunction with a membrane-associated mating pair formation (MPF) that provides a mating channel (Smillie, et al., 2010). These functions where already explained in the previous section.

Apart from the set of genes described above, plasmids are considered to be modular, since they usually contain discrete regions of genes clustered together in functional groups that are responsible for various aspects of maintenance and propagation. Plasmids usually consist of a backbone of selfish modules (figure 5) regularly accompanied by genes coding origins of replication or transfer, centromere sites for plasmid segregation, etc.; along with the backbone there is a block containing a mosaics of translocative and operative elements that encode beneficial traits for the host cell. These elements can include virulence factors that enable colonization of eukaryotic cells, protection against antimicrobial or heavy metal substances or the ability to metabolize certain carbon sources. The adaptive regions of a plasmid do not usually interfere with the normal order of backbone modules (Bailey, et al. 2005, Norman, 2009, Smillie, et al., 2010).

The arrangement of essential plasmid genes inside the backbone modules is very compact and makes them highly vulnerable to insertion of translocative elements, which could compromise the functionality of indispensable genes. The presence of insertion sequences within the adaptive regions of a plasmid backbone usually also provides multiple sites where additional insertions can happen without interfering with overall plasmid functionality. Functional plasmid modules are still susceptible to genetic recombination events or to allelic replacements, provided that the overall functionality of the plasmid is not compromised (Norman, 2009).

In order to be maintained and ensure their own survival, plasmids copy themselves via replication modules (rep), which must ensure that replication proceeds according with the growth cycle of the host cell and without allowing copy numbers to reach a level that imposes an unreasonable metabolic burden on the host. On the contrary, if plasmid copy numbers fall below a certain level, it would lead to plasmid-free (cured) segregants or allow other resident plasmids a competitive advantage. To avoid this, rep modules are often coupled with regions (cop) that ensure the maintenance of stable copy numbers. This usually means that if copy numbers drop below the desired level, plasmids replicate more than once a cell cycle, while if copy numbers are too high plasmid replication stops almost completely (Bailey, et al. 2005, Norman, 2009).

Smaller plasmids can be found in excess of hundreds of copies, while conjugative plasmids are typically found in low copy numbers, which reflects the selective advantage of minimizing the metabolic burden on the host (Norman, 2009).

There is a correlation between high transfer rates and stable maintenance of conjugative plasmids, since where communities in which horizontal transfer rates are high, plasmid conjugation could perpetuate the formation of large communal gene pools, allowing beneficial traits to be shared among the population only restricted by the host ranges of resident plasmids. Hence, a large communal pool might lead to an overall decrease in genome sizes, assuming that only strictly host essential genes would be required within the private pools. On the contrary, if transfer rates were to drop below those needed to sufficiently sustain conjugative plasmids, this might drive the communal pool back into the private pools (and chromosomes) of individual cells, perhaps resulting in an increase in the population's genome size (Bailey, et al. 2005, Norman, 2009).

After the conjugation, the entry of a plasmid into the recipient host, now called transconjugant, often results in the cease of expression of several backbone genes because of the absence of control proteins within the new cell. Low concentrations of copy-control molecules, for example, cause a dramatic rise in the initiation of plasmid replication; ensuring a quick recovery of copy numbers in the transconjugant (Norman, 2009, Seoane, 2010).

Given that there are plasmids that cannot self-transfer but which, in the presence of transfer-proficient plasmids, can be mobilized from host to recipient or retrotransferred from the potential recipient back to the host, it is probable that several distinct transfer mechanisms have evolved (Bailey, et al. 2005).

Types of Conjugative Plasmids

The only protein component of the conjugative machinery that is common to all transmissible, i.e., conjugative or mobilizable, plasmids is the relaxase.

Conjugative

A plasmid that codes for its own set of MPF genes is called self-transmissible or conjugative. It contains both MOB genes and a T4SS (MPF genes).

Mobilizable

This type of plasmid uses an MPF of another genetic element present in the cell. Mobilizable plasmids only carry the genetic information necessary for relaxosome formation and DNA processing, and therefore do not provide the burden of pilus synthesis.

Non-Mobilizable

They are neither conjugative nor mobilizable. They spread only by natural transformation or by transduction.

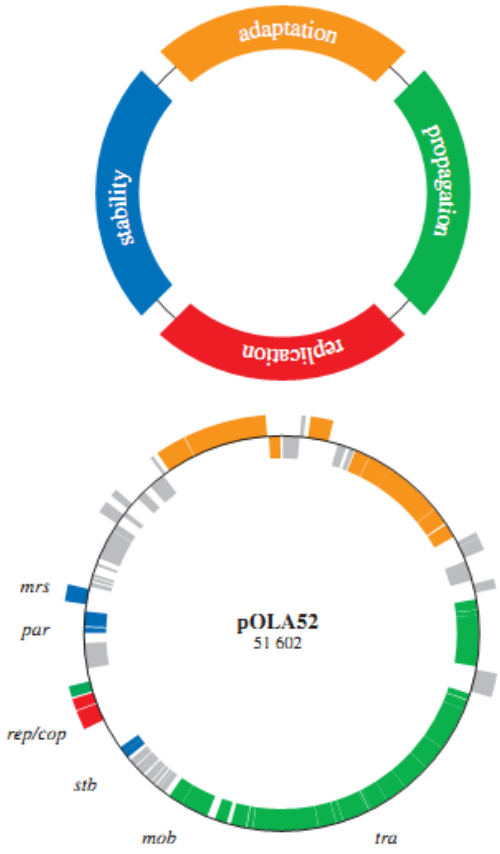


FIGURE 5- AN OVERVIEW OF THE ORGANIZATION OF AN ARCHETYPICAL CONJUGATIVE PLASMID COMPRISED FOUR GENE MODULES: STABILITY (BLUE), REPLICATION (RED), PROPAGATION (GREEN) AND ADAPTATION (ORANGE). GENES ENCODING UNKNOWN FUNCTIONS OR FUNCTIONS NOT DIRECTLY RELATED TO THE FOUR MODULES ARE INDICATED WITH GREY. A. NORMAN, ET AL "CONJUGATIVE PLASMIDS: VESSELS OF THE COMMUNAL GENE POOL,"

Chapter 4 iDynoMiCS

iDynoMiCS (individual-based Dynamics of Microbial Communities Simulator) is an open source simulator that follows an Individual-based Modeling (IbM) approach or agent-based models, where bacteria is individually simulated as a sphere of variable size in a continuous, three-dimensional space, its aim is to describe the different characteristics defining a bacterial population as well as the biology and interactions occurring between individual bacterium and the single cell level (Lardon, et al., 2011).

The purpose of iDynoMiCS is to simulate the growth of populations and communities of individual microbes (small unicellular organisms such as bacteria, archaea and protists) that compete for space and resources in biofilms immersed in aquatic environments. IDynoMiCS seeks to understand how individual microbial dynamics lead to emergent population- or biofilm-level properties and behavior (Lardon, et al., 2011).

iDynoMiCS' IbM framework is based on the same principles of cellular automata models, that is substrate uptake, metabolism, maintenance, cell division and death. The concentrations of substrates and products (i.e. oxygen, carbon sources, ammonia, nitrite, or nitrate) are obtained from diffusion and reaction. By using a 3D space, the movements of the simulated bacteria can have the same degree of freedom as in reality, without using a predefined grid and no global laws such as exponential population growth are applied. iDynoMiCS is also based on other software such as BacSim and Phobia (Lardon, et al., 2011).

Some of the various functional characteristics that iDynomics has to offer are:

- Capability of simulating biofilms in 2D or 3D.
- A pressure field that allows the shrinking or consolidation of the biofilm.
- EPS simulation and continuous excretion.
- A stochastic chemostat model.
- Able to predict and describe the dynamics of bacterial growth
- Plasmid conjugation
- Multiple tools for post-processing simulation results like POV-Ray, Matlab or R scripts

IDynomics is open source software released under a GPL-like CeCILL license that allows modification and redistribution of the software. It is programmed in JAVA and the input files are in XML format, the results will be outputted in xml and a zip file containing POV-Ray files to render 2D or 3D images (Lardon, et al., 2011).

Class Structure

Entire projects with many different simulations may be performed without any extra programming of iDynoMiCS because all parameters of the system, solutes, and the microbes can be configured from a parameter file (called protocol file) in XML (eXtensible Markup Language) format. Programming is only necessary when new boundary conditions, new kinetic equations, or species with new behavior are introduced.

The object-oriented structure of iDynoMiCS means that each major piece of the model is self-contained and may easily be updated or replaced with a similar class that offers different functionality. In addition to enabling modularity, the object-oriented structure also allows more complex classes to derive from simpler ones that won't require complexity. For example, agents in iDynoMiCS are made up of a hierarchy that increases complexity at each stage: first existence, then component masses, then reactions, then location and size, and finally any species-specific behavior.

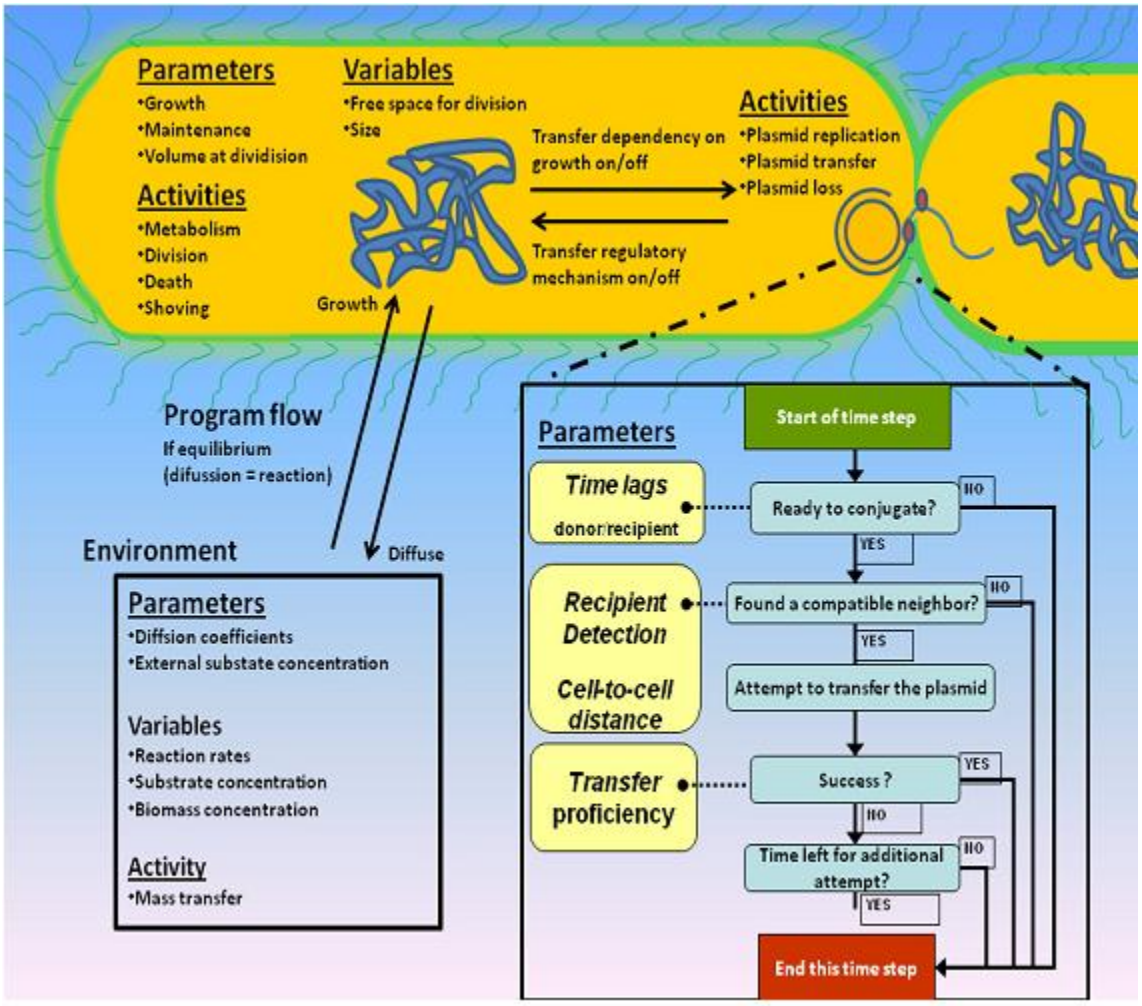


FIGURE 6 - IDYNOMICS MODEL STRUCTURE OVERVIEW. TWO MAIN MODULES MAY BE DISTINGUISHED: CELL INTERACTIONS WITH THE ENVIRONMENT (LEFT HALF) AND PLASMID REPLICATION AND TRANSFER (RIGHT). J. SEOANE, "INDIVIDUAL-BASED ANALYSIS AND PREDICTION OF THE FATE OF PLASMIDS IN SPATIALLY STRUCTURED BACTERIAL POPULATIONS: PHD THESIS,"

Packages

iDynoMiCS is divided in several packages separated by functionality to ease maintenance and updates. In the next figure we can see the hierarchy of the different packages.

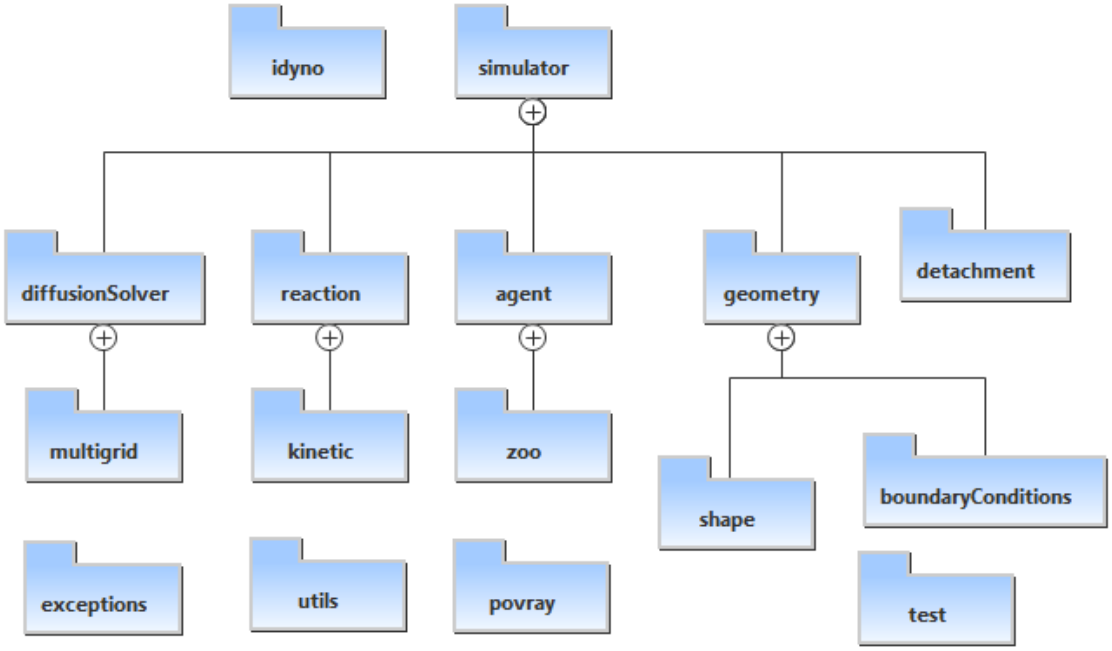


FIGURE 7- PACKAGE DIAGRAM OF IDYNOMICS.

Below is a brief overview of the packages functionality:

- iDyno: This package contains the main class IDynomics and the class SimTimer that is responsible for maintaining the timer in the simulation.
- Simulator: It is the main package where almost all basic operative classes reside, like the world definition and the Agent Container to mention a few. It has multiple sub packages:
    - Agent: The Agent package contains the basic definition, high level methods and common functionality to all simulated microbes.
        - Zoo: In here we can find the definition and individual behavior to each different type of bacteria.
    - Detachment: This package contains classes pertaining to detachment of biomass from the grid.
    - DiffusionSolver: Classes in this package are the responsible of maintaining the environment, some of its classes are responsible for: calculating pressure, chemostat and the multigrid.
    - Geometry: The classes related to shape and vectors are allocated here. This package has two sub packages:
        - Boundary Conditions: Classes that define the different types of boundaries in which the grid can be defined.
        - Shape: In this package are the different geometrical shapes in which the bacteria will be represented.
    - Reaction: In here all classes responsible for the calculations of the biomass are stored.

    - Kinetic: This package stores the different kinetic solvers available.
- Exceptions: This package contains the class that handles the custom iDynoMiCS exceptions.
- PovRay: This package is used to store the routines that generate the POV-Ray input files at the end of the simulation.
- Utils: This package holds different classes that are unrelated to the main objective of the simulator. In here we can find logs, zip generation classes, different math functions, the random number generation, etc.

Main Classes

In this section several of the most important classes will be explained, since there are over 90 classes in the iDynoMiCS software, describing them all is out of the scope of this project. In order to better understand the modifications done to achieve our main goal, we will focus on the classes relevant to the creation of bacteria and the plasmid-conjugation mechanism (figure 9), afterwards we will briefly go through the simulation steps in an attempted pseudo-code-like run. This will help us understand how things in iDynoMiCS are managed. It is to be noted that all parameters come from the Protocol File, an xml that is read upon initialization of the simulation.

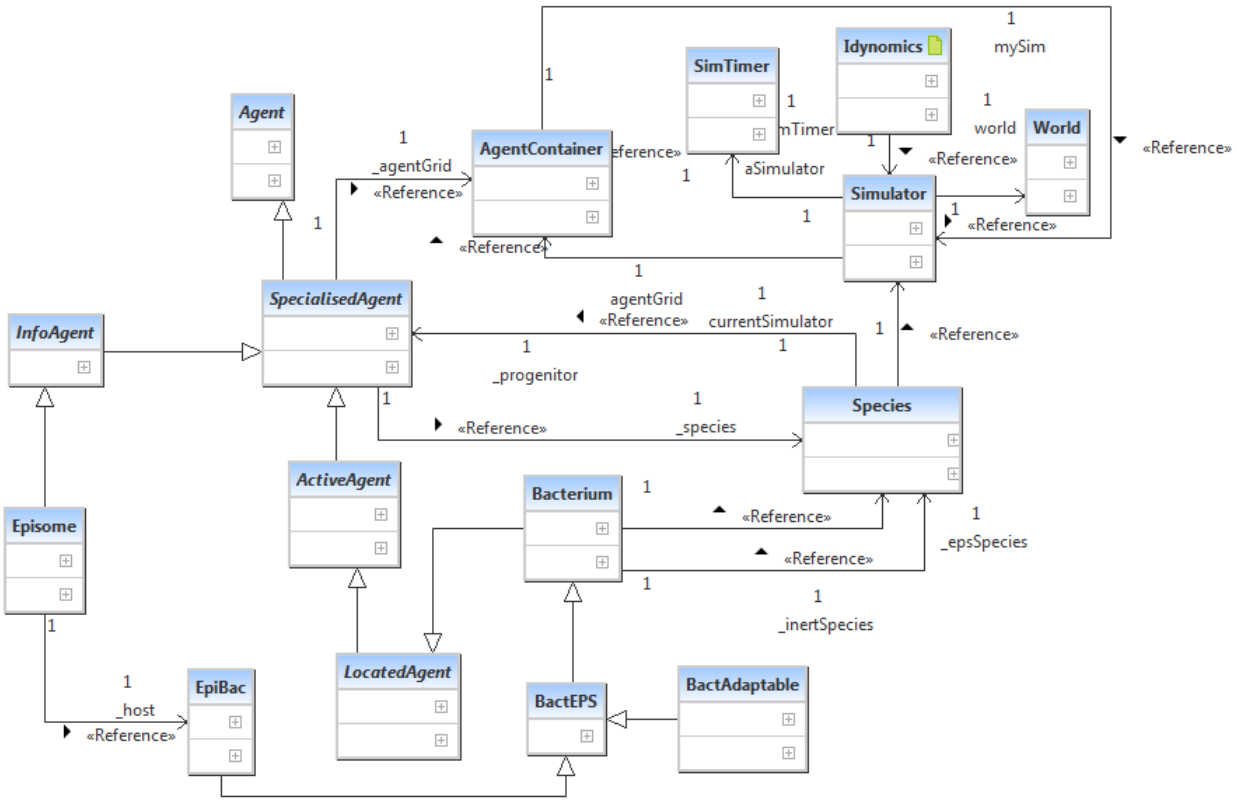


FIGURE 8- A FRAGMENT OF THE CLASS DIAGRAM OF IDYNOMICS SIMULATION SOFTWARE. ONLY THE CLASSES RELATED TO THE SIMULATION OF PLASMID CONJUGATION PRIOR TO ANY MODIFICATION ARE SHOWN.

- **IDyno**: Is the main class of the program, this class prompts the user for the protocol file that can be for a new run or a previous one. This is where everything starts.
- **Simulator**: This class manages the high level functions, such as the instance of the environment, the main instance of the timer, etc.
- **World**: The world class does just that. It sets the world and high level environment parameters for the simulation such as the bulk(s) and domain(s).
- **SimTimer**: It manages the system iterations and in-simulation time.
- **AgentContainer**: This class is where the individual instances of the simulated bacteria reside; also it controls the behavior of all agents in the simulation. This class is in charge of the processes that all the bacteria undergo, like suffling, death, shoving, etc.
- **Agent**: The Agent class is where the existence of the bacterium is first defined, is the parent class from which all the different behaviors, reactions, etc. are built upon.
- **SpecializedAgent**: Inherits from the Agent class, and gives the bacterium (agent) its species.
- **ActiveAgent**: this gives the agent its basic parameters and methods, such as growth rate and volume rate, it also bounds the reactions.
- **LocatedAgent**: This class sets the parameters of where in the world it is located and other physical data. It also controls the actions of the agent, like growth, division and death.
- **Bacterium**: Up until this point, an agent could be any type of agent, like an EPS for example. This class gives the agent the status of bacterium and confers some extra parameters.
- **BactEPS**: this class inherits form the bacterium class and gives the agent the functionality to excrete EPS particles.
- **EpiBac**: In this class is where the individual behavior of the bacterium type is defined. Specifically this class creates a bacterium that can hosts plasmids. In here is also the functionality for the conjugation mechanism.
- **Episome**: This class called when an instance of a plasmid is required by the EpiBac class.

The simulation class performs these actions at each simulation step:

- Apply new step in system time.
- Create new bacteria if possible and remove dead ones.
- Resolve diffusion and pressure reactions.
- Calls AgentContainer to go through the individual time steps of every agent.
- Write result file for current step.

AgentContainer class:

- Shuffle the list of active agents according to a random seed.
- Check whether the elapsed time is less that the value defined in the global step time: if so do the following tasks:
- Adjust local time step
- Calculate and apply pressure movement on all agents
- Run the step method for all agents

- Shuffle agent list again
- If using a chemostat, the system will flush a specific percentage of agents
- Remove dead and misallocated agents from list
- Solve spatial spreading. That is running the shoving algorithm for all allocated agents according to a computed movement value.
- Recalculate shoving process between agents.
- Update agent status
- Identify the borders of the biofilm
- Determine erosion and shrink elements on the border

LocatedAgent class:

- Calculate the growth rate of the agent
- Update radius and volume
- If the agent is ready, segregate.
- Update the status of the agent.
- If applicable, inform of agent death.

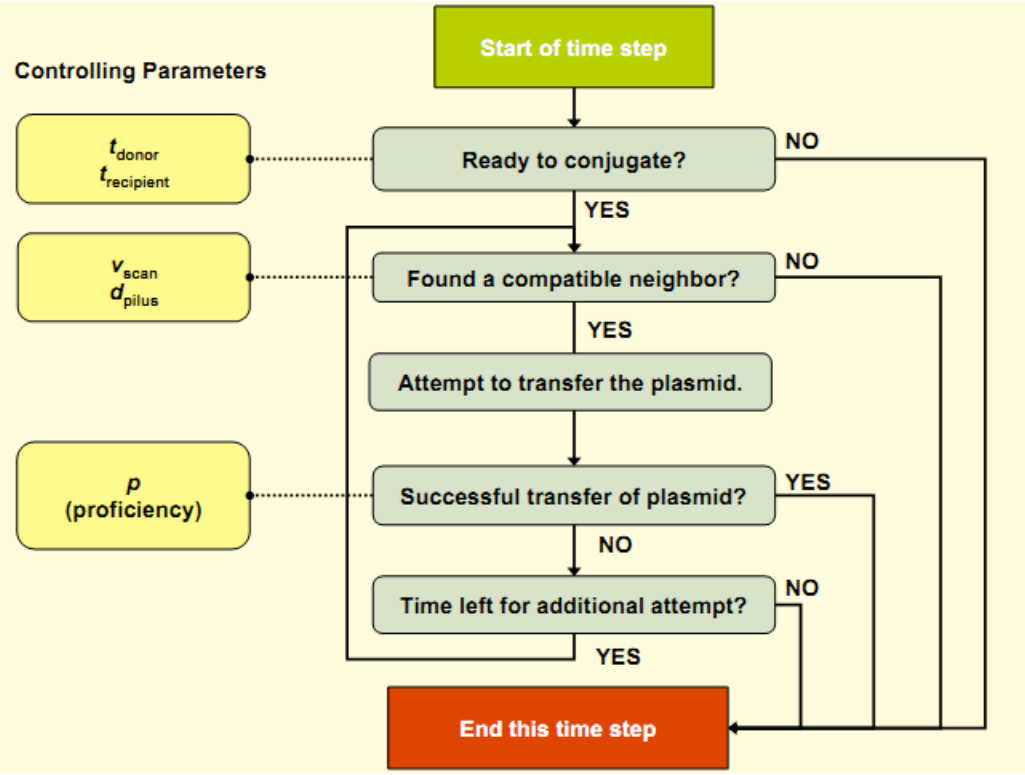


FIGURE 9- ALGORITHM RULING CONJUGATIONAL PLASMID TRANSFER IN IDYNOMICS. J. SEOANE, "INDIVIDUAL-BASED ANALYSIS AND PREDICTION OF THE FATE OF PLASMIDS IN SPATIALLY STRUCTURED BACTERIAL POPULATIONS: PHD THESIS,"

EpiBac class, this class makes it possible for the agent to host plasmids and other EPS:

- Check if plasmid still exists
- Updates sizes and volume
- Manages EPS, if it is ready to excrete it, do so
- Divide or die if aplicable
- Conjugate if there is a recipient cell near, the last exchange is bigger than the exchange lag and other conditions are met. (figure 10)

Episome class, this class doesn't have any routines to do for every time step the simulation takes, but it was considered important to explain some of the parameters this class has:

- Copy number: the total number of copies this plasmid has in the host
- Pilus length: the length of the conjugative pili, this will set the host range in which the bacterium can search a viable receptor
- Last reception and last exchange counters: time counters storing when the last reception of a plasmid or the last exchange was. They will be compared against lag counters to see if it can donate or receive a plasmid
- Reactions: this list stores reactions tied to the plasmid
- Repressed state: Boolean marking whether or not this plasmid is repressed. It accounts to the whole plasmid since there are no individual genes to express.
- Compatibility Marker: It indicates whether or not a foreign plasmid can coexist with this one. If the new plasmid has the same compatibility marker it means they are not compatible.
- Loss probability and transfer proficiency: The loss probability indicates if the plasmid will keep on living inside the daughter cells. The transfer proficiency is the one that indicates how often this plasmid can be transferred.

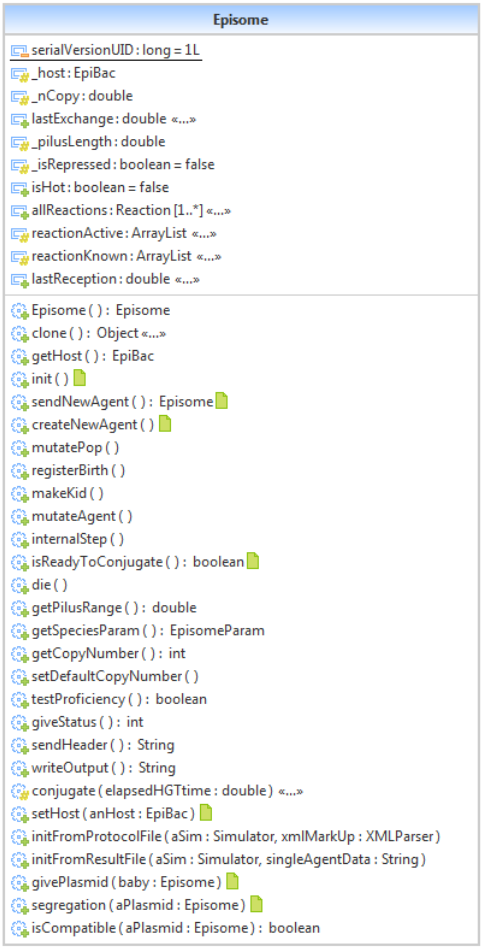


FIGURE 10 - EPISOME CLASS UML DIAGRAM

Both EpiBac and Episome have an additional class each that is in charge of reading the necessary data in the protocol file upon creation, these classes are called EpiBacParam and EpisomeParam respectively.

Output Files

iDynomics generate several zip files as the result of the simulation, each one for every time step ran:

- Agent_State, Agent_State_Death: This files record the location, mass, radius, and other properties specific to the agents for each iteration time. The Death files record which agents and of which species died during the iteration.
- Agent_Sum, Agent_Sum_Death: Record the total population, mass and growthRate for every discticnt bacterium. The Death files record the same properties of the total of deaths.
- Env_State, Env_Sum: this files hold the information regarding the environment for every timestep.
- Log files the simulation record for the duration of the simulation.
- POV-Ray files to render images of every time step.

Part 3 Proposed Solution

Chapter 5 Adaptations to the simulation process

This chapter will cover the description of the changes made to iDynomics in order to simulate plasmid particles and conjugation. In order to complete the objective several modifications to existing files and additional classes were created.

The following points where identified as operational requirements, based off on the need for a more robust plasmid conjugation framework. These points will be explained in detail in the following sections. All required input will be configured in the protocol file.

Operational Requirement

1. A bacterium can hold one, many or none plasmids of any size
2. A plasmid can have one, many or none origins of transmission
3. A plasmid can have one, many or none T4SS machinery
4. A plasmid can have one, many or none genes encoded, including MOB genes.
5. A plasmid without T4SS can use another plasmid T4SS as long as it is compatible and its expressed in the sharing plasmid
6. A gen may express only one protein
7. A gen must have a promoter zone
    a. The promoter zone of a gene may activate or inhibit the expression of the gene
    b. The promoter zone may or may not have a logical condition regarding the existence of any proteins within the bacterium
8. A plasmid may need any or all of the following proteins for its MOB genes:
    a. Relaxase
    b. Nickase
    c. T4CP
    d. Custom logic condition regarding the existence of protein within the bacterium
9. The proteins expressed coexist inside the host. They have a degradation time.
10. The concentration of a specific protein will change with the time, depending on how many genes inside the host are expressing it, or whether it lifetime has ended.
11. Proteins have a minimum and maximum concentration thresholds within the host, if the concentration of a protein is lower than the minimum, it will be considered inexistent. On the contrary, the concentration of a protein cannot go higher than the maximum threshold.
12. Stochasticity must be ensured

Redefining the Plasmid Model - Overview

The plasmid model in iDynoMiCS, as previously mentioned, is a very simple one, containing only the relevant methods to ensure the continuation and transmission of the plasmid to other agents.

iDynoMiCS is already prepared for an EpiBac agent to hold more than one plasmid, since they are stored in a list. However, the Episome class functions as an indivisible individual, which defeats the modular nature of a plasmid; therefore, to ensure modularity and ease of implementation, the plasmid was divided in two parts concerning functionality: the first part, the backbone, relates to the defining of the transfer machinery as shown in figure 12; this part will contain the necessary items required for conjugation. The second is where all the genes are encoded into the plasmid.

The backbone module is where all the necessary information for the plasmid main operation is encapsulated, that is: the Origin of Transfer, T4SS and MOB genes. This section will only incorporate the required elements as a list, there will be no actual genes encoded here. This will allow the simulation of plasmid types; if in this section the plasmid lacks any of the substances expressed by genes mentioned earlier, and it can't find the required substances within the bacterium, the plasmid becomes a non-mobilizable plasmid and can only be transferred via segregation. On the contrary, if the bacterium has all listed substances in enough concentration the plasmid is considered a mobilizable-conjugative plasmid, whether or not the plasmid's own genes are the ones expressing the substances.

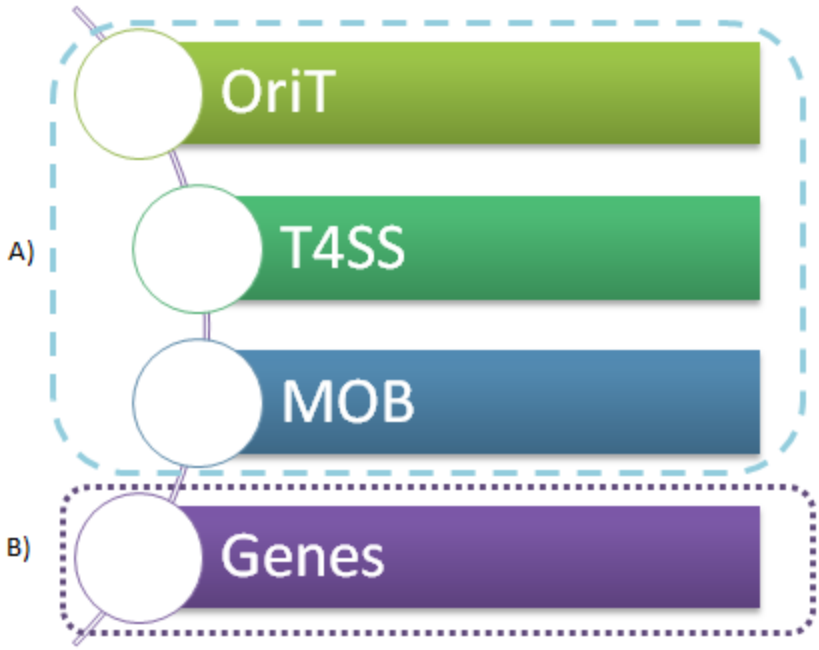


FIGURE 11- MODULAR DIVISION OF THE EPISOME CLASS. A) ENCAPSULATES THE MODULES NECESSARY TO THE OPERATION AND CONJUGATION OF THE PLASMID. B) OTHER GENES THAT ARE ENCODED IN THE PLASMID.

All the MOB required substances will be treated as proteins, with the exception of oriT and T4SS. For otiT, only its existence in the plasmid will be checked, since is one of the first conditions to be checked while in conjugation. T4SS have to be encoded and expressed in a gene, but will not be conditioned to a certain concentration threshold. Also, a plasmid can have one or more oriT and T4SS configured.

Each gene has a promoter zone that designates a logical condition related to the concentrations of proteins within the host. It can sense if one or various substances exists or are missing, and inhibit or activate the expression of the gene if the conditions are met. Also, the backbone module may require a special condition to be met, this will be treated the same as the inhibition-activation function.

All the substances that are expressed through the genes are stored in the host. This means that the EpiBac class will now have the task of maintaining the concentrations of substances in its interior. For this, a special container class was designed, in which all the individual proteins are managed, similar to iDynoMiCS own AgentContainer class. The verification of the conditional will be done by another class. With this all the mayor operative requisites are met.

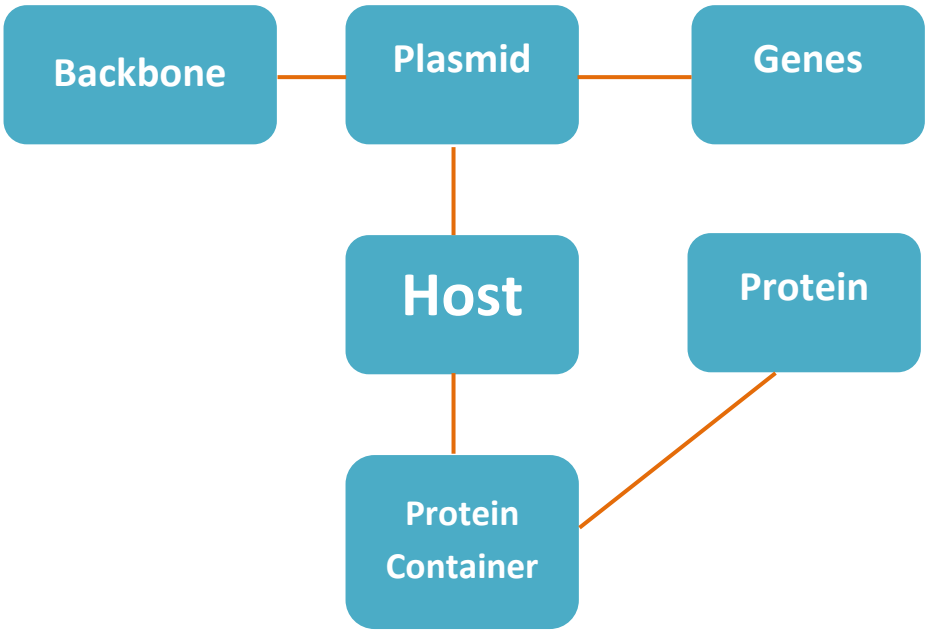


FIGURE 12 NEW PLASMID-CONJUGATION CONCEPTUAL MODEL

Exclusions

The following are the compromises that had to be made in order to maintain the simulation within reasonable computational requirements.

- The copy number of a plasmid will not be taken into account, since it will be too processor-heavy to have an individual instance of a plasmid for every copy there is.

- Considering that there will be no more than one copy per plasmid in the host, the processes for copy number maintenance will not be simulated either.
- Neither the compatibility aspects once the conjugation is completed, regarding the competitive advantage one plasmid may have over another sharing the same hosts. In this sense only entry/surface exclusion will be simulated.

UML diagram of the new model

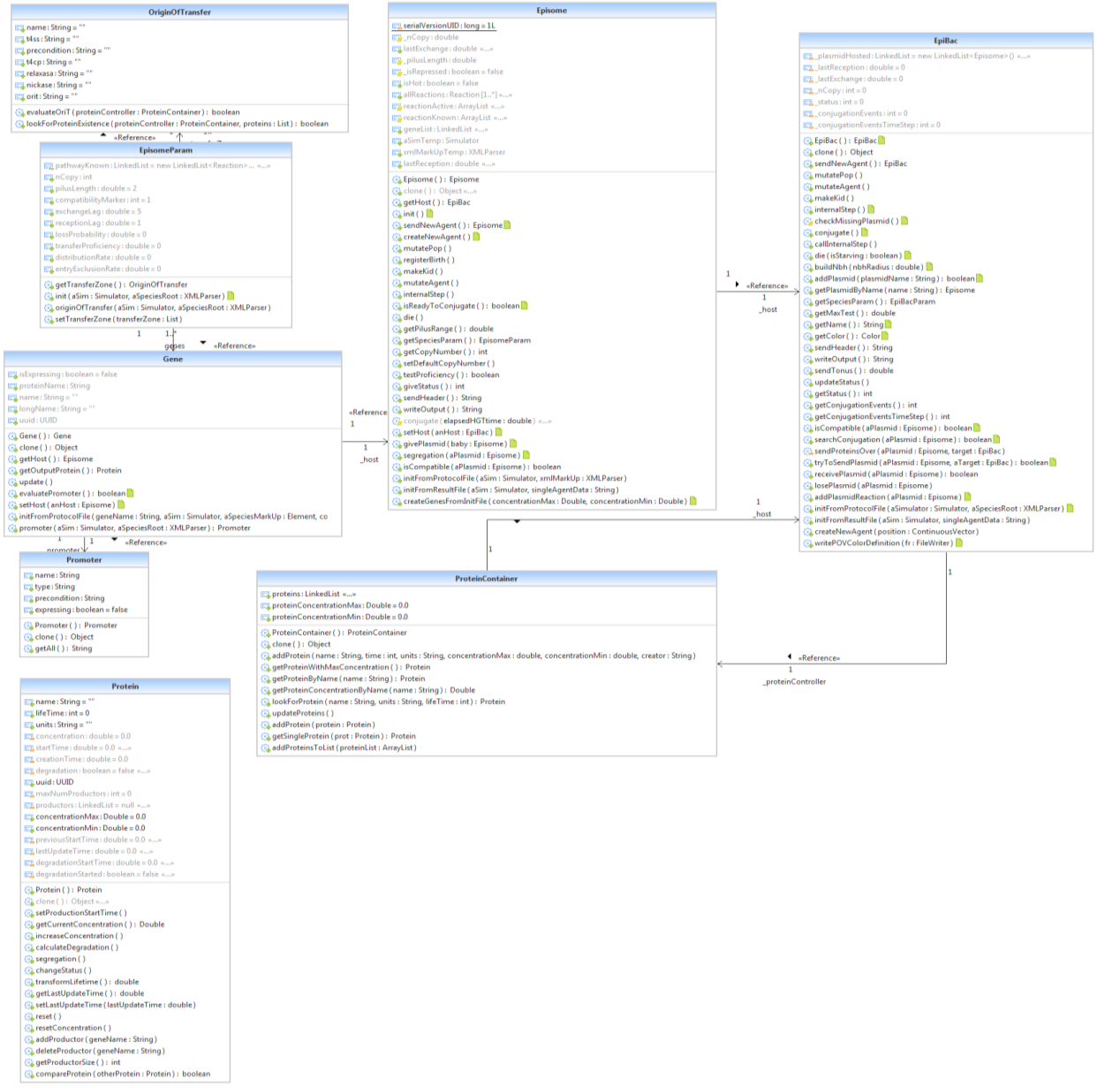


FIGURE 13 SEGMENT OF THE UML DIAGRAM. SHOWING ONLY THE MODIFIED SECTION.

This is the final UML class diagram of the new model; all the new classes were added in a package called extension inside simulator.agent package. The classes shown in the diagram are explained below:

- EpiBac – Host class, this class will be the one managing plasmids, proteins and conjugation.
- Episome – Class hosting the plasmid, manages genes encoded in it.
- EpisomeParam – This class extracts the needed information from the protocol file and sets the required variables.
- OriginOfTransfer – This class stores the backbone part of the plasmid-model, is where the conditions for conjugation are stored
- Gene – represents a gene encoded into the plasmid
- ProteinContainer – manages the proteins that exists within the hosts
- Protein – represents a type of protein in the bacterium, its concentration varies with time.
- Parser – this class parses the promoter and backbone conditions with the existences of protein
- Promoter – in this stored the condition that regulates the gene's expression

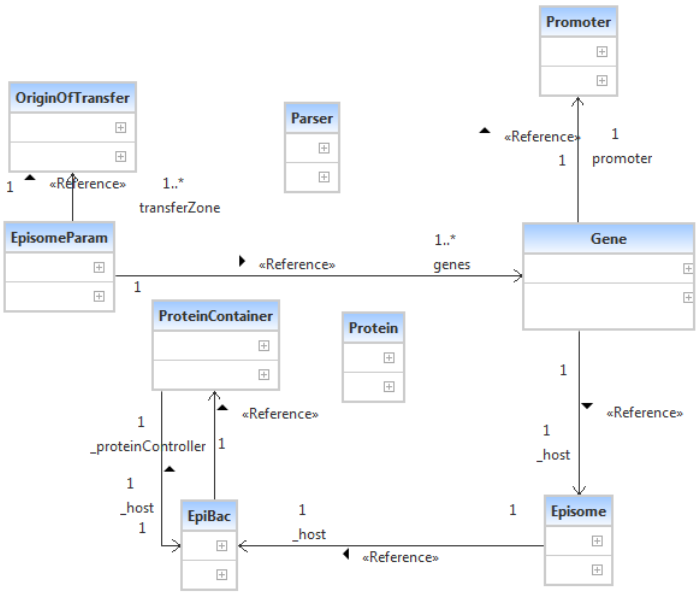


FIGURE 14-SHORTENED VERSION OF UML DIAGRAM TO SHOW RELATIONSHIPS BETWEEN CLASSES

Initialization File

The following xml extract is a segment of the protocol file in which the plasmid MyPlasmid is configured:

```
<species class="Episome" name="MyPlasmid">

      <!--  iDynomics required fields   -->
      <param name="pilusLength">22</param>
      <param name="exchangeLag">1</param>
      <param name="receptionLag">1</param>
      <param name="lossProbability">0.005</param>
      <param name="transferProficiency">10</param>
      <param name="compatibilityMarker">1</param>
      <param name="nCopy">1</param>
```

```
            <!-- extension fields   -->
            <param name="distributionRate">15.0</param>
            <param name="entryExclusionRate">15.0</param>

            <!—Conjugation zone, can be multiple or one or none, remember to declare
which T4SS is the one that can move the OriT -->
            <conjugationZone name="zone1">
                  <param name="OriT">MyOriT</param>
                  <param name="T4SS">T4SSk</param>
                  <param name="Relaxasa">MyRelaxasa</param>
                  <param name="T4CP">T4CPi</param>
                  <param name="precondition">MyProtein</param>
            </conjugationZone>

            <!—One or more genes can exist. If the genes required in the conjugation
      zone are not in this list, another plasmid will need to encode them -->
            <gene name="MyGene">
                  <param name="longName">gene 1</param>
                  <promoter name="MyPromoter">
                        <param name="promoterType">inhibitor</param>
                        <param name="precondition">MyProtein2</param>
                  </promoter>
                  <param name="outputName">MyProtein</param>
                  <param name="outputTime">15</param>
                  <param name="outputUnit">Minutes</param>
            </gene>

            <gene name="MyRelaxasa">
                  <param name="longName">MyRelaxasa</param>
                  <promoter name="promoter">
                        <param name="promoterType">inhibitor</param>
                        <param name="precondition"></param>
                  </promoter>
                  <param name="outputName">MyRelaxasa</param>
                  <param name="outputTime">20</param>
                  <param name="outputUnit">Minutes</param>
            </gene>
      </species>
```

We can divide the code into 3 main sections; the first is where the required input for the correct function of the plasmid is set, is explained in the following table:

Name	Description
pilusLength	Defines the distance in which the host can find a suitable receptor
exchangeLag	The minimum time that must pass before transferring another plasmid
receptionLag	The minimum time that must pass before receiving another plasmid
lossProbability	The probability the plasmid has of being lost during segregation
transferProficiency	The probability of the plasmid being transferred
compatibilityMarker	Plasmids with the same value can’t coexists in the same hosts, so entry is denied
nCopy	Maximum number of copies of the plasmid inside the cell

The second section, the conjugation zone, specifies the substances required for conjugation and an optional precondition. This section could not exist, making it a non-mobilizable plasmid, or it could have more than one with different substances. All information encoded in this section will be treated as proteins, with a few exceptions, and may or may not be set within the protocol file. If a substance is declared here that does not have a gene expressing it in the next section, the plasmid will look for the existence of the substance within the host. There may be plasmids that require more or few substances for conjugation.

The OriT is a special case, since in reality is more of a section encoded in the plasmid backbone, the system will only check if at least one conjugation zone block has this line within the protocol file. If it is missing, the plasmid will be treated as non-mobilizable even if it does have a conjugation zone.

For T4SS, that is, the conjugation machinery that builds the philus, if configured, will only check that it is expressing at the time of conjugation.

The conjugation zone section is set to receive the following 5 variables: OriT, T4SS, T4CP, relaxasa and nickase plus a precondition that allows a simple logical function regarding the existence of any other proteins that may be available.

The final section is where the actual genes are “encoded” into the plasmid. The field longName was devised to make the identification easier when reading the log file. OutuputName is the name of the protein that will be expressed by the gene, while outputTime and outputUnit are the maximum lifetime that a single protein will survive inside the host, the units can be seconds, minutes or hours, but the suggested value is the timespan a bacterium takes to segregate.

Inside the gene is the promoter element, its presence is required or the simulation will fail. In here we set up the type of the promoter, inhibitor or activator, and a precondition necessary to the activation or inhibition of the gene’s expression. This precondition is a simple logic formula regarding the presence or not of one or more proteins in the host cell, for this, the outputName of the genes are used. If the gene has no precondition it means it will never be met, for example, in the MyRelaxasa gene above there is no precondition present for the inhibitor promoter zone. This means that this gene will never stop expressing the MyRelaxasa protein.

Two different genes can’t have the same name; conversely, these two genes can express the same protein, as this will only lead to a faster concentration rise within the host bacterium.

Adapting iDynoMiCS

Initialization

The first modification was made in the Episome class. In the initialization routine, were information regarding the plasmid is read. During this procedure the class EpisomeParam is called, inside it, a new OriginOfTransfer instance is created and put into a list for every conjugationZone element that exists, along with other variables required for the correct function of the plasmid. Afterwards, the reactions are set.

Once the initialization of the plasmid is made, it's added to a list containing all plasmids that belongs to that host cell inside the original EpiBac instance. Immediately after, another method is called within the episome instance that reads the protocol file and subsecuently creates the genes and proteins and add them to another list, inside the plasmid class for the genes and for the proteins in the host EpiBac class. The proteins are created with a concentration of 0, and the genes are not expressing. Once all the bacteria and plasmids are created, the EpiBac class is cloned until the initArea number is met for every bacterium. The following diagram shows the initialization process.

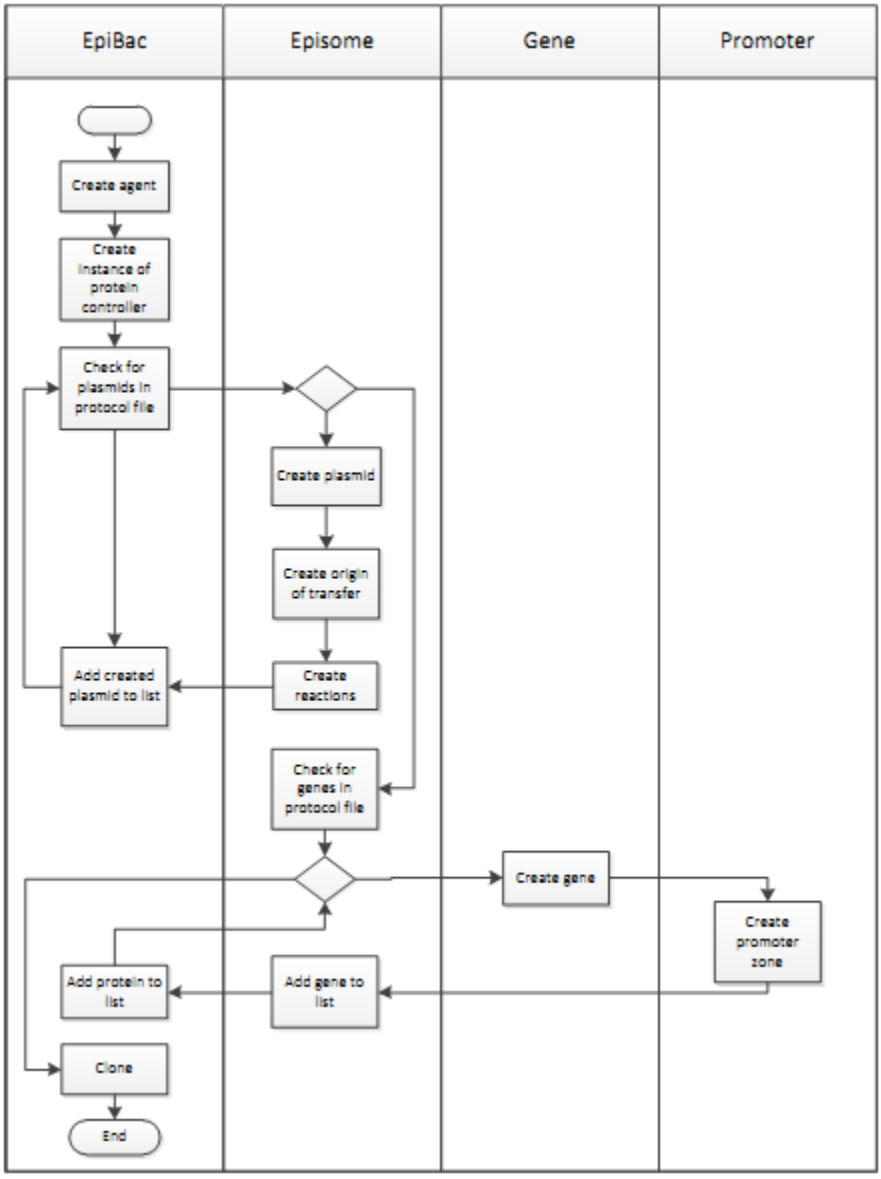


FIGURE 15-INITIALIZATION PROCEDURE DIAGRAM

Simulation Timestep

For every time step the simulation class calls, there is a set of routines to be run, such as pressure calculations, resolving reactions and updates within the bacteria. Every type of bacteria overrides the

agent class internalStep method to accommodate changes unique to its species. In this case, some modifications done to the EpiBac class were done to accommodate the new genes and proteins.

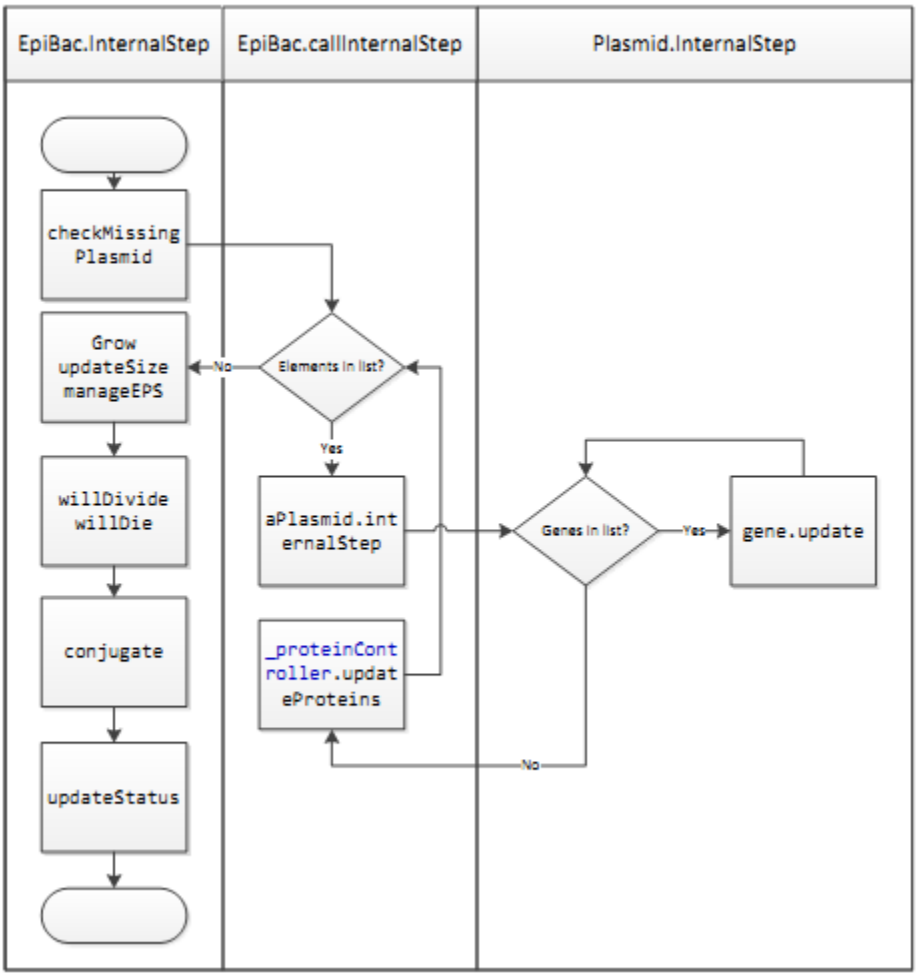


FIGURE 16-INTERNAL STEP PROCEDURE DIAGRAM

To facilitate the analysis, first we will see the overall modifications to a higher level and afterwards we will see certain important parts in more detail. In EpiBac class the internal step method is exactly the same as the original code, however, the method called callInternalStep (figure 16) traverses the list of available plasmids and call each plasmid's internalStep method which in turn updates every gene's status. That is to say, every gene's promoter condition is matched against the current protein concentrations inside the host bacterium. If true this condition can start or stop the expression of the gene. In Figure 17 we can appreciate the decision tree diagram in which the expression of the gene is settled, once the decision path is taken, the gene will locate its protein through the ProteinController class and change the expression flag.

This process takes part inside the update method in the gene class (figure 16). The actual matching of the proteins concentration and the parsing of the logic function contained in the promoter zone is done by the support class Parser.

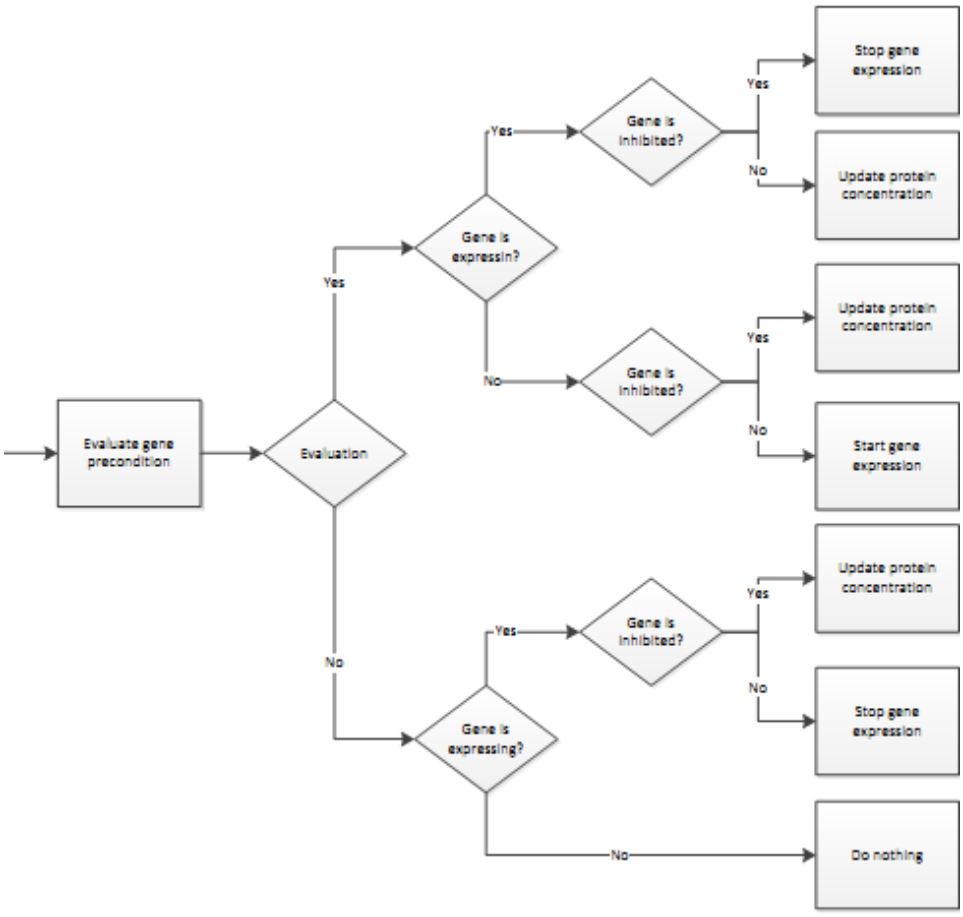


FIGURE 17-GENE.UPDATE DECISION TREE

Immediately after all the plasmids have updated its genes, the method callInernalStep in the EpiBac class calls the routine for protein updates, this ensures that the new protein concentrations will be set according to the changes in gene expression.

All protein interactions are managed by the ProteinController class; it is a type of interface that makes easy the interactions between hosts and the substances that live inside it by treating them as a sole existence, while still retaining the capacity to control each protein separately. This class contains a list of the different proteins that will, at some point, exist within the cell. Also, they are never deleted, since a protein can cease to exist during a period of time but can increase its concentration once a gene starts expressing it again.

ProteinController has as well the responsibility to make available to the protein instance relevant information from the simulator, provide access to the proteins, as well as filtering and searching.

Once the appointed time has passed from the first expression of the gene, the protein will start to degrade, even if a gene or genes are expressing it still. This will lead to a slow downgrading of the concentration of a protein over time until no genes are expressing the protein anymore. There is also a set minimum of the protein concentration to be taken into account. It does not matter if there are still some proteins inside the hosts, since they will be so few to be detected by the plasmids. The degradation mechanism helps maintain the balance of the proteins within the hosts.

Once the proteins and genes are updated the system will continue with the flow of the internalStep method in the EpiBac class, as seen in Figure 16. The methods grow and updateSize in charge of the development of the bacterium over time, manageEPS is self-explanatory. The methods willDivide and willDie are actually decisions that will trigger the dividing and/or dying routines of the cell, both will be explained below while the conjugation method has its own section.

If method willDivide is true, it calls LocatedAgent's divide, which will clone the agent and everything inside it, including plasmids, genes and proteins, mutate some of the values for the other daughter and reset both agents to the size and status of newly born bacteria. The clone method will set the protein concentrations in half and the genes expressing status will remain the same. During this process, the LossProbability value set in the protocol file determines if a plasmid will be copied to the daughter cells or if it will disappear from the hereditary line. WillDie, if true, will "kill" the agent and unsubscribe it from the AgentController class, taking with it reactions and its plasmids.

Conjugate Method

The conjugate method is the one that decides whether or not a plasmid is transferred. It is divided in two separate routines and several different parts. This method starts by traversing the list of available plasmids, for each plasmid it checks whether or not the plasmid can be transferred. If it can, then the second routine is put into motion, and where the final checks before the actual transfer are met.

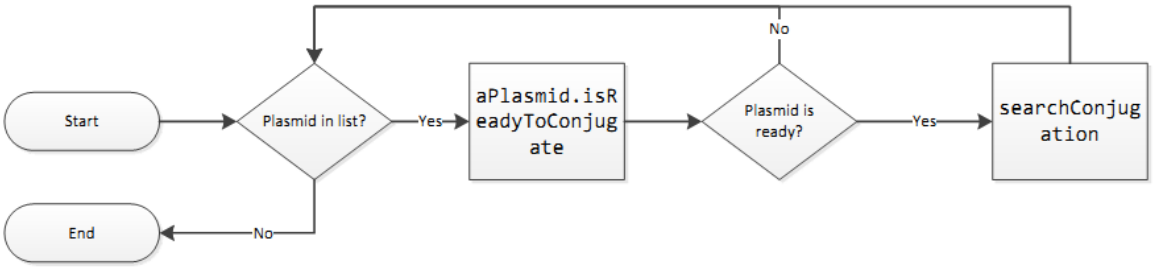


FIGURE 18 – CONJUGATE METHOD FROM EPIBAC CLASS. IT CALLS ISREADYTOCONJUGATE FOR EVERY PLASMID REGISTERED

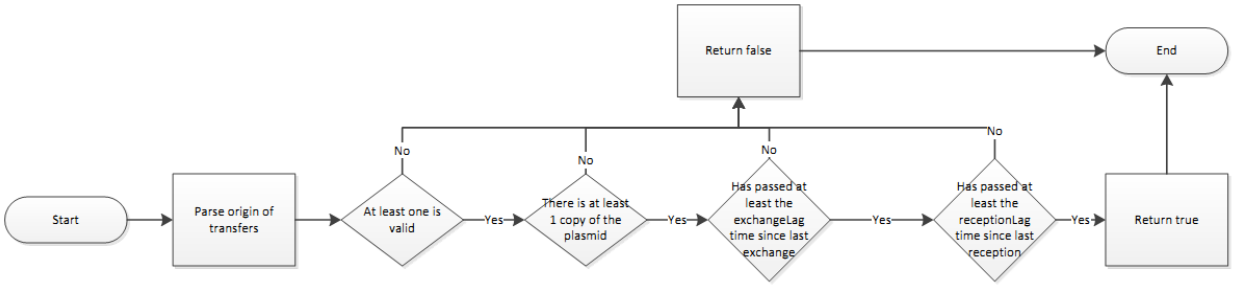


FIGURE 19 – ISREADYTOCONJUGATE METHOD FROM EPISOME CLASS. RETURNS TRUE IF FIT FOR TRANSFER

In figure 18 we can see the flow diagram for the isReadyToConjugate procedure called during the conjugate method in EpiBac class. As previously said, a plasmid can have multiple origin of transfer zones. In this case, this class encapsulates all the necessary items for conjugation, in the same way the conjugation zone of the protocol file does. In this sense, each instance of the OriginOfTransfer class will have a distinct OriT value, as for the other required variables, such as relaxasa, T4SS, etc., would not matter if the same values were used as long as they exists within the hosts.

If during the execution of the isReadyToConjugate method at least one instance of OriginOfTransfer is considered to be true, then the following comparisons are tested to ensure the plasmid availability to transfer:

- All conditions in the conjugation zone have been met
- There is at least 1 copy of the plasmid
- The time passed since the last exchange of the plasmid is greater than the exchange lag value.
- The time passed since the last reception of the plasmid is greater than the reception lag value.

As soon as all these points have been verified, the plasmid is subsequently approved for conjugation.

Once selected a candidate for transfer the next method, searchConjugation, will arrange for a receptor and settle the missing steps in the bacterial conjugation process.

The only parameter the method searchConjugation (Figure 19 – A) receives is the selected plasmid, which is the used to obtain the pilus length and build the hosts neighborhood, whit this, all bacteria within range can be a possible receptor, as such; they are stored in a list. Afterwards, iDynoMiCS will calculate the number of attempts at conjugation the host bacterium can make. This number will be reduced by one every time an attempt is made, and will undertake every opportunity the hosts gets until this number reaches 0. During each iteration a bacterium is chosen randomly from the list as a possible receptor, it is then passed along the previously elected plasmid to the method tryToSendPlasmid (Figure 19 – B).

The routine mentioned above will check that the to-be transferred plasmid is compatible with the ones already inside the receptor, along with calculating the probability for the success of the transfer (transfer proficiency).

Once these two conditions are met, and the relevant information sent to the system log, the reference of the selected plasmid is then sent to recipient's recievePlasmid method, where it is cloned and subsequently registered.

The last step in the conjugation process is done once the transfer is successful; consisting on declaring the new proteins in the recipient's proteinController.

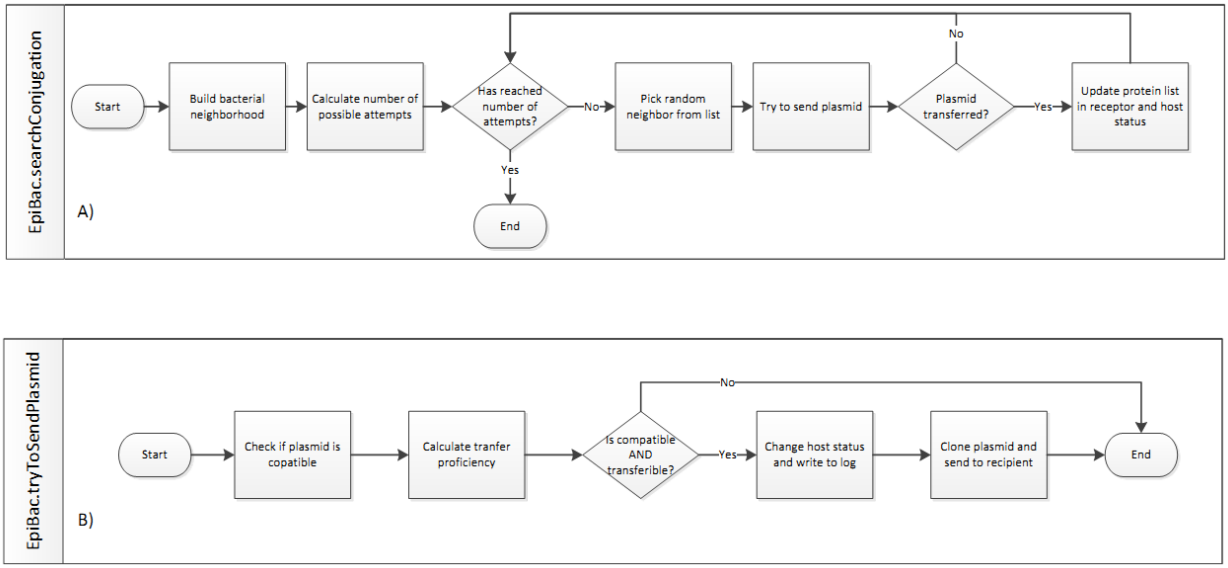


FIGURE 20 - A) SEARCHCONJUGATION AND B) TRYTOSENDPLASMID METHODS FROM EPIBAC. THE PLACE WHERE THE ACTUAL TRANSFER OF THE PLASMID IS DURING B), WHICH IS CALLED IN THE STEP BEFORE THE LAST CONDITION IN A) THROUGH EACH ITERATION OF THE LOOP

The method called searchConjugation will select potential receptors from the neighbor list randomly and try to conjugate with them until the attempt counter reaches 0. This is to ensure the transfer rate given at the beginning.

With these modifications, the new conjugation model for iDynoMiCS was implemented and working, the modular nature of the software allowed for changes to be, for the most part, seamless. However it should be noted that a great deal of how the system behaves depends solely on the data configured in the protocol file.

Part 4 Simulation Results

Chapter 6 Model Validation

In order to assess the newly made enhancements, several experiments were designed; which will be explained in the next chapter. First off, it was necessary to ascertain that the output was as accurate as possible to real results, even though, in this kind of setting, the configuration of the protocol file plays a major role in the overall results of the simulation. To achieve the desired fidelity, real data is needed.

Conjugation in biofilms, even though is been known to happen for years

For this purpose, the data available by Irene Del Campo, et al. in "Determination of conjugation rates in solid surfaces" was used as the real-life model. The experiments done in del Campo's work were made over petri dishes and did not form a biofilm, but as explained by del Campo: "the bacterial cells are already in close contact at the beginning of the mating experiment, forming at least a one-cell layer on the agar surface. […]These high cell concentrations seem to be essential for the close contacts needed for conjugation to happen at optimal rates." Close contacts between cells, as well as minimal shear forces are the principal requirements of conjugation, since both these features are characteristic for biofilms; it has been supposed that this data is suitable for comparison (Król, et al., 2013)

For the simulation, Del Campo's experiment assumptions were maintained; namely they assume that (1) mating occurs randomly, (2) plasmid loss by segregation is negligible, (3) all bacterial cells have the same growth rate (approximately 50 min) and (4) transconjugants are not able to transfer the plasmid to new recipient cells (Del Campo, et al., 2011)

The simulation was carried as follows: A 1 µm long *E. coli* custom rod model was used (iDynoMiCS only had sphere looking bacteria models available for use), the simulation space was a 3D grid whose width and height both correspond to 32 µm. At the vertical edges of the available space, the bacteria wrap to the opposite side of the volume. The initial configuration is of a single layer of cells on the floor of the volume strewn randomly. Nutrients were in a well-mixed liquid and constant, ensuring all bacteria have equal access to them.

The smallest time unit in the simulations is one minute. The e. coli divide after a time period decided by a Gaussian distribution with a mean of 50 minutes, and a standard deviation of 4 minutes. When the model e. coli divide a shoving algorithm previously adapted to by Arteaga (2012), was utilized.

This experiment assumes that bacteria can conjugate at most twice between divisions (an estimate based on discussions with the supervisor of this work). Conjugation procedure explained in chapter 4.

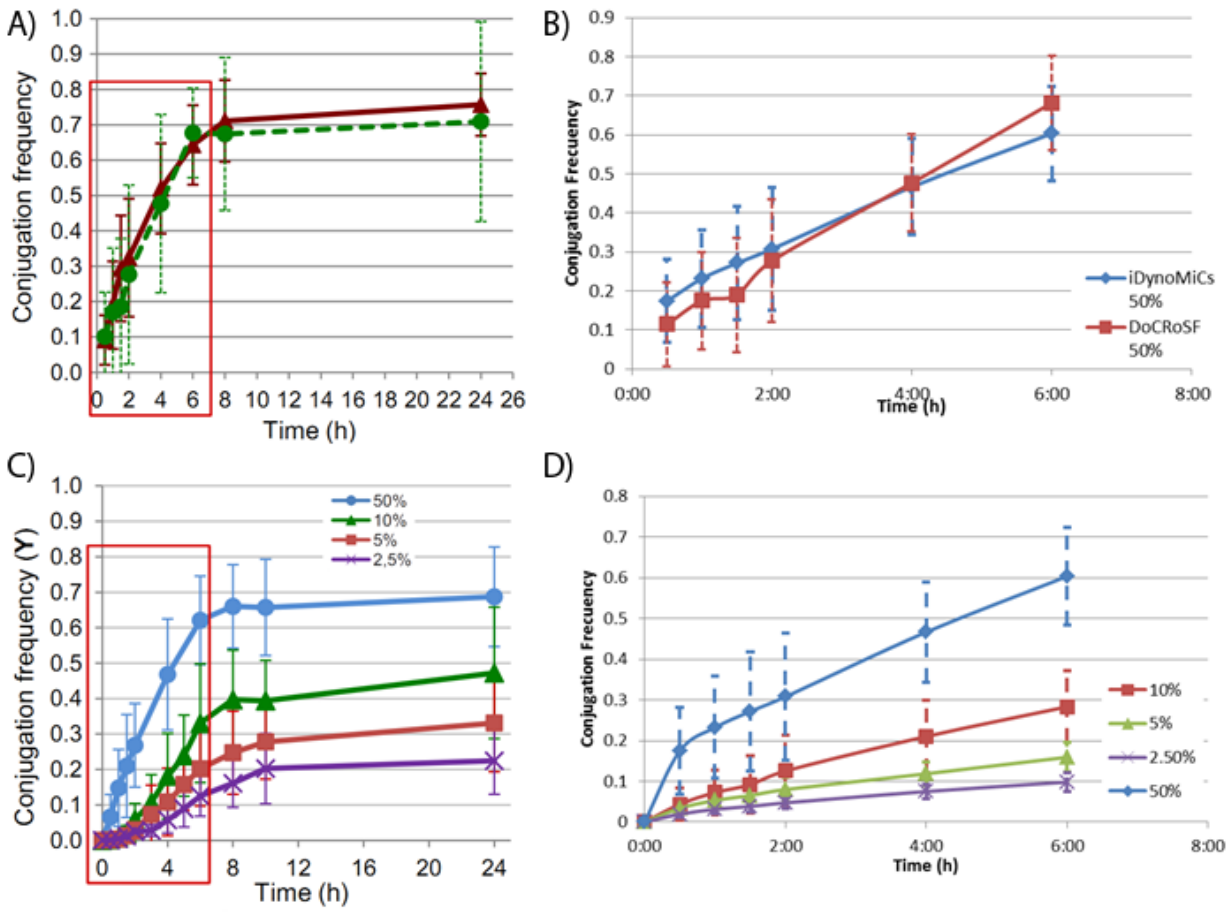


FIGURE 21 - A) DEL CAMPO'S COMPARISON OF CONJUGATION FREQUENCIES AS ESTIMATED BY FLOW CYTOMETRY AND REPLICA-PLATING. B) COMPARISON BETWEEN A)'S REPLICA-PLATING (RED) AND IDYNOMICS SIMULATION RESULTS (BLUE), BOTH WITH 50% OF DONORS DURING THE FIRST 6 HOURS OF THE EXPERIMENT. C) DEL CAMPO'S GROWTH OF DONOR CELLS ON THE SURFACE OF AGAR PLATES AS CALCULATED BY REPLICA-PLATING FOR DIFFERENT D/R RATIOS. D) IDYNOMICS RESULTS FOR DIFFERENT D/R RATIOS. THESE CLOSELY RESEMBLE THE FIRST 6 HOURS OF DEL CAMPO'S EXPERIMENTS. DEL CAMPO, ET AL, "DETERMINATION OF CONJUGATION RATES ON SOLID SURFACES"

The measurement unit was the conjugation frequency, also known as conjugation rate, and several other names. Different approaches have been used to determine the efficiency of plasmid transfer, Del Campo, et al, refers as transfer frequency (γ) to the efficiency of plasmid transfer in the overall population, that is, the ratio of the number of transconjugants (T) to either the number of donors (D) or of recipients (R). The formula used to calculate the frequency of conjugation was $\gamma = T/(R + T)$.

Due to computation and time restraints, the time of iDynoMiCS simulations were kept at 6 and half hours, 10 simulations were made for each Donor/Recipient ratios which were 2.5%, 5%, 10% and 50%, a total of 40, of which, each one has slightly different configuration parameters to ensure true, non-deterministic, unique runs . Figure 20 A and B graphs show the similarity in the conjugation rates between the set of simulations and the expected results from del Campo's work with a difference of less than 10%. Likewise the graphs C and D, the first is from Del Campo's experiment showing the

conjugation rates for the different D/R ratios and its evolution over time, the former, is iDynoMiCS experiments that likewise show a similarity close enough to Del Campo's experiment during the first 6 hours. With this demonstration we can assume a correct configuration of the system.

Chapter 7 Experiment Design and Results

The AND circuit Sensor

As a proof of concept, a small experiment featuring and AND sensor was designed to be performed once the system was correctly calibrated with real data. The purpose of this experiment was to measure the sensibility at detecting two signals present in a bacterial population and triggering a response. In this case, the response was in the form of a green fluorescent protein (GFP). Two scenarios were designed for comparison: a classic AND sensor (fixed); non-mobilizable AND plasmid, expresses a GFP once both input signals are transferred to the host. And a Mobile AND sensor; mobilizable AND plasmid that conjugates once both input signals are transferred to the host while also expressing the GFP encoded inside.

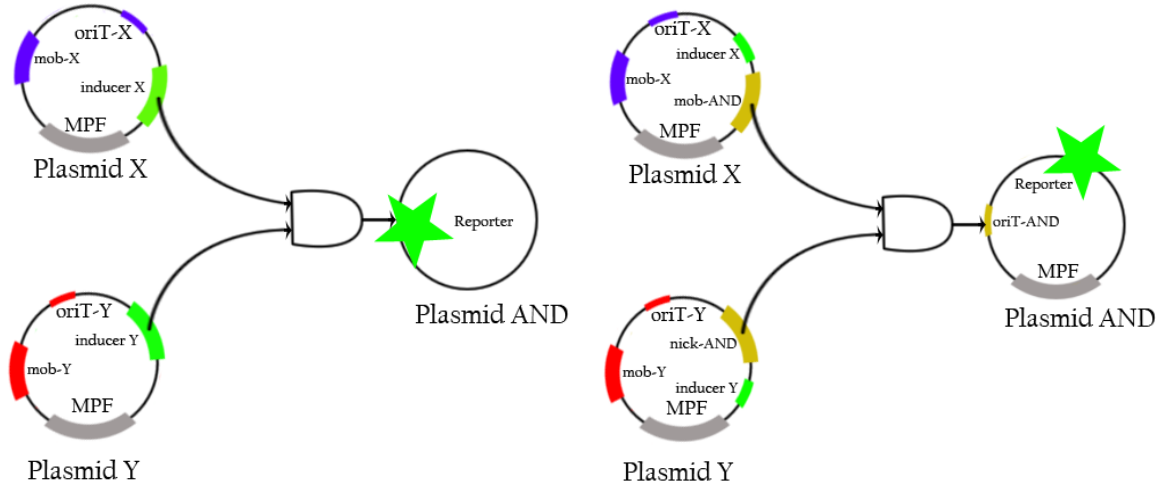


FIGURE 22 - LEFT- FIXED AND SENSOR, BOTH INPUTS (X,Y) ON THE SAME HOST TRIGGER GFP'S EXPRESSION. RIGHT- MOBILE AND SENSOR, THE INPUTS CONFER THE AND PLASMID WITH THE MISSING CONJUGATIVE COMPONENTS AND TRIGGER THE GFP'S EXPRESSION.

In a more detailed description: both scenarios hold the same structure and functionality, the two signals (X,Y) are conjugative, there is a large chunk of the population that is plasmid-less. Both starts with the same initial population, as well as D/R ratios (2.5, 5, 10, 25, and 50%), furthermore 10 simulations per ratio were made. Each simulation, 100 in total were run for 6 and half hours each.

To simulate scenarios in which inputs are few and far between, donor ratios were further divided in 3 to accommodate both inputs and sensor plasmids. This resulted in, for example, the case of 10% donors: 3.3% will be comprised of input X, 3.3% of input Y, 3.3% of AND plasmids, and 90% of empty recipients.

This very basic AND circuit can be extended and adapted for a variety of uses, such as detecting cancer or other diseases. The reasoning behind the setup of this experiment is to ascertain which amount of sensors are needed to correctly detect the desired inputs in a mixed population, as well as to determine if a mobile sensor, more complex than its counterpart, is worth the extra metabolic burden in order to ameliorate sensitivity.

Results

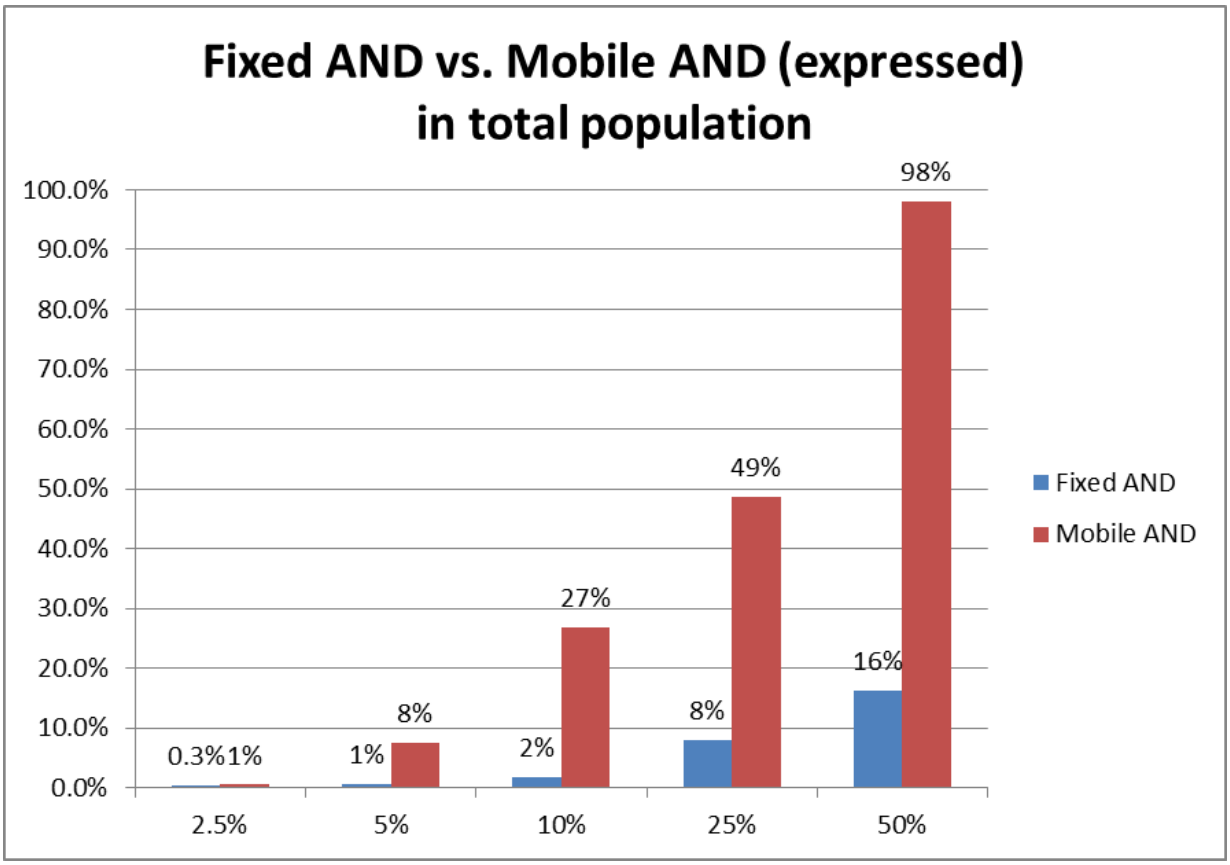


FIGURE 23 - INVASION OF THE EXPRESSED SENSOR IN THE POPULATION. AND SENSOR IS ONLY ACTIVATED IF BOTH INPUTS ARE BEING EXPRESSED INSIDE THE SAME HOST. ALSO, IN MOBILE AND'S CASE (RED), BOTH INPUTS CONFER THE MISSING MOBILITY TO THE SENSOR'S PLASMID.

Overall, the results obtained were as expected. The second scenario, mobile AND, has a considerable increase in sensibility, as shown in figure 22. In average, mobile AND case, was able to detect both inputs 30% better than its fixed counterpart. This is due to the advantages offered by HGT, since classic AND sensor can only be reproduced by segregation it takes more time to find a host with any of the two inputs present. And while the host with the fixed AND sensor can conjugate both inputs, the rate at which fixed AND sensors distributes in the population is too slow.

In cases where the ratio donor/recipients is too low, ex: 2.5% (that is a 0.83% of the initial population for each input and sensor) a mobile AND sensor does not make a huge difference, since both inputs are needed to transfer the sensor to other recipients, the probability of finding those inputs is low but slightly better than fixed AND. In average, simulations running with this d/r ratio reach 40% input invasion by the last hour. It is highly probable that, if the simulations continued, a suitable detection rate were to be obtained.

On the other extreme, in the case where the donors equals half the initial population (50% - 16.6% of each input and sensor) we can see a total occupation of plasmids in the biofilm. Figure 21 shows us only the percentage of expressed sensors, that is 98% of the population has the two inputs plus sensor inside

and they are most likely overburdening the whole system. Meanwhile fixed AND sensors were only able to detect 16% of the inputs within the same 6 and half hours. And while it is a good detection percentage, the total of input occupation within the population reached 95% in average by hour 4.

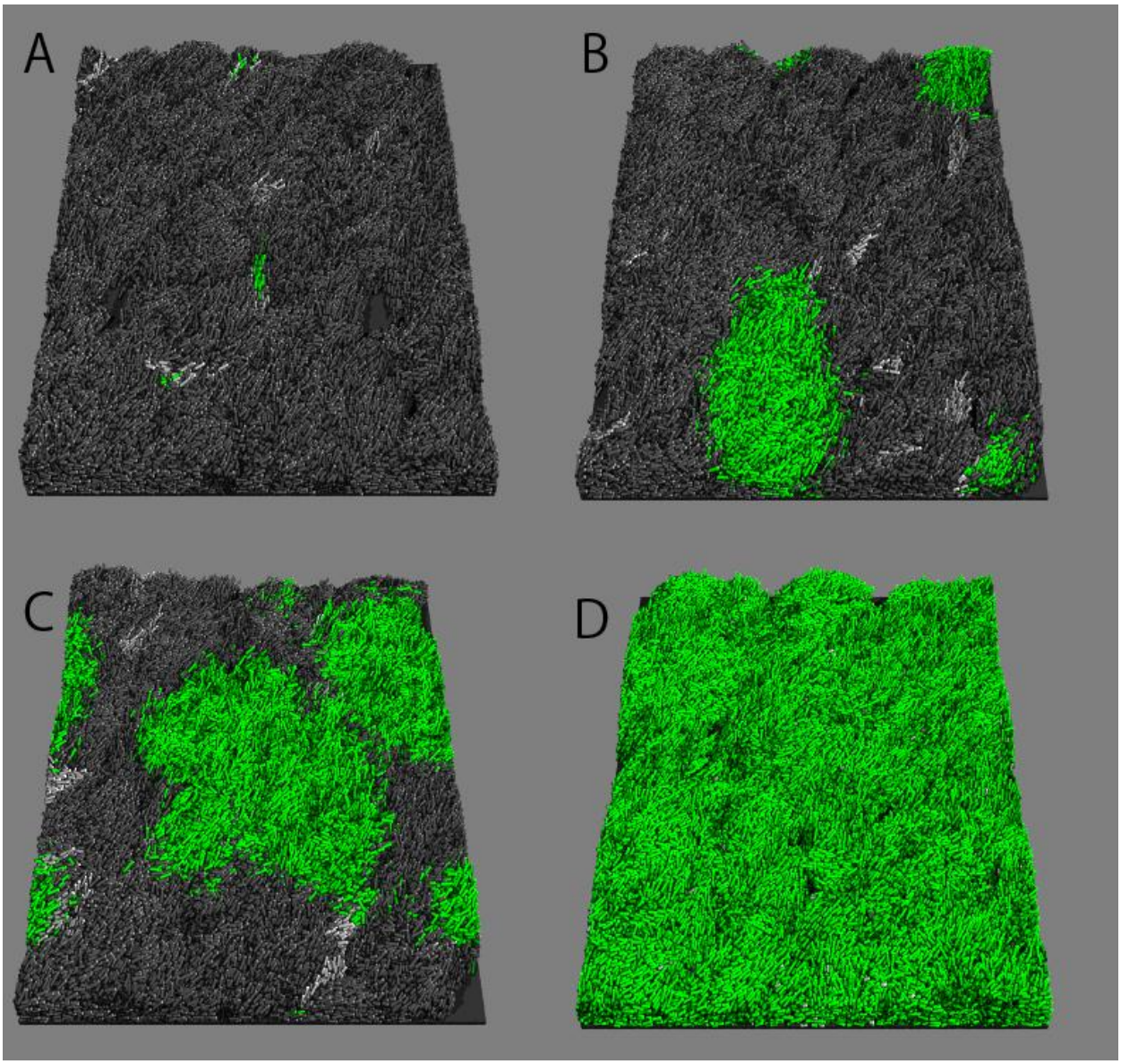
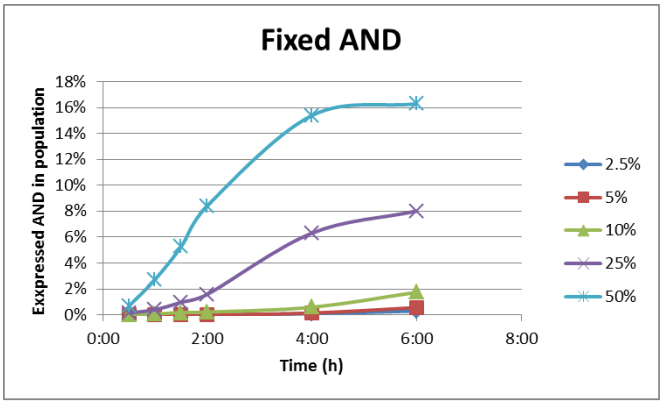


FIGURE 24 - RENDERED IMAGES OF MOBILE AND SENSOR SIMULATIONS AT HOUR 6. A) D/R RATIO 2.5%. B) 5%. C) 10% D) 50%

The best results so far were obtained with d/r ratio of 10% (3.3% for each input and sensor).Inputs in this setting never really manage to infect the whole population, though they are close, averaging a 90% of occupation by hour 6. AND plasmid in the population averages to a 33%, of this total 82% is expressed, equaling to 27% of the population being fluorescent. Unlike other cases, the system is not overly seized by plasmids and a third of the input signals are detected.



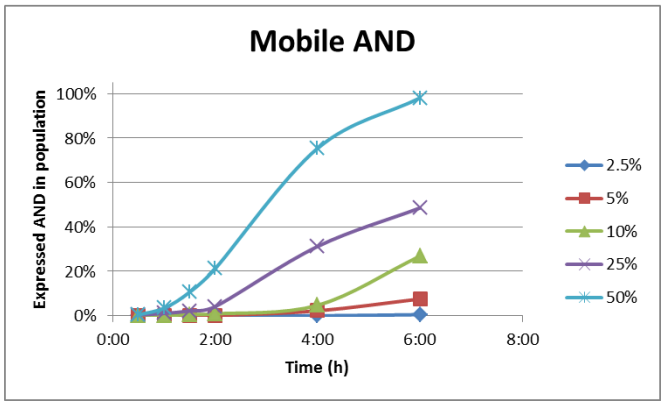


FIGURE 25 - EXPRESSED AND SENSOR INVASION IN POPULATION OVER TIME. LEFT- FIXED AND SENSOR IS ONLY ABLE OF VERTICAL REPRODUCTION (SEGREGATION) WHILE MOBILE AND (RIGHT) IS CONJUGATIVE ONCE THE TWO INPUTS ARE TRANSFERRED TO THE SAME HOST.

The development of the expressed AND sensors are shown in figure 24 contrasts both scenarios over time. It is easy to see that the more inputs within reach, is easier for the sensors to pick them up, but also affects the number of sensors available.

The advantages mobile sensor against a fixed one is primarily its sensibility. Being capable of converting additional hosts into sensors is certainly a good leverage, likewise, the detection range increase, the opportunity to discern inputs farther away and its low detection time makes this type of sensor superior to its simpler counterpart. On the contrary, this advantage comes at the cost of added metabolic burden, which, if taken to extremes can cause the host to not conjugate at all. The fixed AND sensor may be slower but if left enough time it will certainly yield the desired results.

Chapter 8 Conclusions and future work

Conclusions

Plasmids most of the time confer bacterial populations a set of extra evolutionary advantages, on other occasions, plasmid functionality is relegated to adjacent processes that are not directly needed for survival. In any case, plasmids and horizontal gene transfer have been studied extensively during this last two decades, in addition to biofilms, since they represent the most common form of microbial life in natural environments and are a hotspot of conjugational plasmid transfer. Studies focusing on plasmid conjugation in biofilms remain scarce due to the difficulty in obtaining accurate data mostly because of problems with quantitative detection and good model systems (Del Campo, et al., 2011, Król, et al., 2013).

IDynoMiCS is a framework dedicated to individual-based modeling of microbial communities, e. g. biofilm growth. Its processes are based on individual interaction between bacteria and can easily perform many different types of simulations. iDynoMiCS is an open source project, thus under constant development. The modular nature of this software allows for seamless modifications and enhancements to be run on top of core processes without needless meddling.

The development of new modeling tools and the enhancement of the previously existing ones, that can address the spatial and biological similarity typically associated with biofilms, offer the possibility to get new insights into this problem. However, the lack of reliable estimates available makes very difficult the implementation and further validation of the model.

That said, under the need for a more factual plasmid conjugational model, this project was envisioned. Due to the lack of accurate conjugation data in biofilms this work is heavily based on the records of the experiments done by Del Campo's (2012) work. Likewise, it also incorporates the previously made adaptations of Arteaga (2012) to implement bacillus morphology into the simulator.

The results obtained from the enhancements done to iDynoMiCS where satisfactory, the implementation objectives were met. That is, developing a reliable, more adjusted to reality plasmid conjugation model. The calibration and validation of the model were particularly laborious, since most of the power of iDynoMiCS comes from the configuration given in the protocol files. In the end, the plug-in like module was able to adequately simulate Del Campo's experiment results with an error of less than 10%. The experiment conducted as a proof of concept also yielded the expected results, that is, a mobile AND sensor is more sensible than a classic sensor of the same type, demonstrating the aptness of the proposed code.

Future Work

Using the current work as basis, there is still a lot to be expanded on. For example:

- Stratification of nutrients, this way we can simulate biofilms growing on agar surfaces, cells grown in the bottom of the biofilm will have more access to nutrients opposed to those in the upper layers.
- Code optimization to reduce simulation time.
- Also, the creation an interface in the gene classes to allow more than two different gene functionalities, right now, genes either produce proteins or fluorescence. However, this could be extended to produce, for example, a self-killing protein, etc.

References

Bailey, M., Hansen, L. H., Kroer, N., Wuertz, S., & Sørensen, S. J. (2005). *Studying plasmid horizontal transfer in situ: a critical review*. Nature Reviews Microbiology. doi:10.1038/nrmicro1232

Del Campo, I., Ruiz, R., Cuevas, A., Revilla, C., Vielva, F., & De la Cruz, F. (2012). *Determination of conjugation rates on solid surfaces*. Plasmids, *67*(2), 174-182. doi:10.1016/j.plasmid.2012.01.008

Król, J. K., Wojtowicz, A. J., Rogers, L. M., Heuer, H., Smalla, K., Krone, S. M., & Top, E. M. (2013). *Invasion of E. coli biofilms by antibiotic resistance plasmids*. Plasmid. Retrieved from www.elsevier.com/locate/yplas

Lardon, L. A., Merkey, B. V., Martins, S., Dötsch, A., Picioreanu, C., Kreft, J. U., & Smets, B. F. (2011). *iDynoMiCS: next-generation individual-based modeling of biofilms.* Environmental Microbiology, 13, 2416-2434. doi:10.1111/j.1462-2920.2011.02414.x

Lipps, G. (2008). *Plasmids: Current Research and Future Trends*. Norfolk, U.K: Caister Academic Press.

Murphy, N. (2012). *Report on simulation of Detection Chain Reaction (DCR)* (D6.1.)

Norman, A., Hansen, L. H., & Sorensen, S. J. (2009). *Conjugative plasmids: vessels of the communal gene pool*. Philosophical Transactions of The Royal Society B: Biological Sciences. doi:10.1098/rstb.2009.0037

Smillie, C., Garcillan-Barcia, M. P., Francia, M. V., Rocha, E. P., & Cruz, F. D. (2010). *Mobility of Plasmids*. Microbiology and Molecular Biology Reviews. doi:10.1128/MMBR.00020-10

Seoane, J. (2010). *Individual-based analysis and prediction of the fate of plasmids in spatially structured bacterial populations: PhD thesis*. Kgs. Lyngby: Department of Environmental Engineering, Technical University of Denmark.

Arteaga, J.P. (2012). *Enhancing iDynoMiCS framework to simulate Rod-shape bacterial colonies growth*.

Appendix

Sample Validation Protocol File

```
<?xml version="1.0" encoding="UTF-8"?>
<!--

#########################################################################
 iDynoMiCS: individual-based Dynamics of Microbial Communities Simulator
#########################################################################

  Website: http://www.idynomics.org

-->


<!-- The entire simulation description is contained within the <idynomics> mark-up tags. -->
<idynomics>

 <!--#####################################################################
  SIMULATOR SECTION
 #####################################################################-->

 <simulator>

  <param name="restartPreviousRun">false</param>
  <!-- <param name="randomSeed">42</param> -->
  <param name="outputPeriod" unit="hour">0.416666667</param>

  <timeStep>
   <param name="adaptive">false</param>
   <param name="timeStepIni" unit="hour">0.416666667</param>
   <param name="timeStepMin" unit="hour">0.025</param>
   <param name="timeStepMax" unit="hour">1</param>
   <param name="endOfSimulation" unit="hour">6.25</param>
  </timeStep>
  <param name="agentTimeStep" unit="hour">0.044166667</param>
 </simulator>


 <!--#####################################################################
  INPUT SECTION
 #####################################################################-->


 <input>
  <param name="useAgentFile">false</param>
  <param name="inputAgentFileURL">agent_State(last).xml</param>
  <param name="useBulkFile">false</param>
  <param name="inputBulkFileURL">env_Sum(last).xml</param>
 </input>
```

```
<!--#########################################################################
 SOLUTES AND BIOMASS TYPES SECTION
#########################################################################-->


<solute domain="MyBiofilm" name="MyCOD">
 <param name="diffusivity" unit="m2.day-1">1e-4</param>
</solute>
<solute domain="MyBiofilm" name="pressure">
 <param name="diffusivity" unit="m2.day-1">0</param>
        </solute>



<particle name="biomass">
 <param name="density" unit="g.L-1">150</param>
</particle>
<particle name="inert">
 <param name="density" unit="g.L-1">150</param>
</particle>



<!--#########################################################################
 WORLD SECTION
#########################################################################-->


<world>
 <bulk name="MyTank">

  <param name="isConstant">true</param>
  <param name="D" unit="h-1">0.6</param>

  <solute name="MyCOD">
   <param name="Sbulk" unit="g.L-1">10e-3</param>
   <param name="Sin" unit="g.L-1">10e-3</param>
  </solute>
  <solute name="pressure">
   <param name="Sbulk" unit="g.L-1">0</param>
   <param name="Sin" unit="g.L-1">0</param>
  </solute>
 </bulk>

 <!-- The computation domain is a physical region that will contain the biofilm,
  and has a more complex setup. -->

 <computationDomain name="MyBiofilm">

  <grid nDim="3" nI="33" nJ="33" nK="33"/>
  <param name="resolution" unit="um">4</param>
  <param name="boundaryLayer" unit="um">40</param>
```

```
   <param name="biofilmDiffusivity">0.8</param>
   <param name="specificArea" unit="m2.m-3">80</param>

   <boundaryCondition class="BoundaryZeroFlux" name="y0z">
    <shape class="Planar">
     <param name="pointIn" x="-1" y="0" z="0"/>
     <param name="vectorOut" x="-1" y="0" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryBulk" name="yNz">
    <param name="activeForSolute">yes</param>
    <param name="bulk">MyTank</param>
    <shape class="Planar">
     <param name="pointIn" x="33" y="0" z="0"/>
     <param name="vectorOut" x="1" y="0" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryCyclic" name="x0z">
    <shape class="Planar">
     <param name="pointIn" x="0" y="-1" z="0"/>
     <param name="vectorOut" x="0" y="-1" z="0"/>
    </shape>
    <shape class="Planar">
     <param name="pointIn" x="0" y="33" z="0"/>
     <param name="vectorOut" x="0" y="1" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryCyclic" name="x0y">
    <shape class="Planar">
     <param name="pointIn" x="0" y="0" z="-1"/>
     <param name="vectorOut" x="0" y="0" z="-1"/>
    </shape>
    <shape class="Planar">
     <param name="pointIn" x="0" y="0" z="33"/>
     <param name="vectorOut" x="0" y="0" z="1"/>
    </shape>
   </boundaryCondition>

  </computationDomain>
</world>



<!--#############################################################################
 REACTION SECTION
#############################################################################-->


<reaction catalyzedBy="biomass" class="ReactionFactor" name="MyGrowthAutotrophs">
 <param name="muMax" unit="hour">0.044166667</param>
```

```
  <kineticFactor class="MonodKinetic" solute="MyCOD">
    <param name="Ks" unit="g.L-1">2.5e-4</param>
  </kineticFactor>
  <yield>
    <param name="MyCOD" unit="g.g-1">-1.5</param>
    <param name="biomass" unit="g.g-1">1</param>
  </yield>
</reaction>



<!--#########################################################################
  SOLVER SECTION
#########################################################################-->
<solver class="Solver_multigrid" name="solutes" domain="MyBiofilm">
  <param name="active">true</param>
  <param name="preStep">150</param>
  <param name="postStep">150</param>
  <param name="coarseStep">1500</param>
  <param name="nCycles">5</param>

  <reaction name="MyGrowthAutotrophs"/>

</solver>

<solver class="Solver_pressure" name="pressure" domain="MyBiofilm">
  <param name="active">true</param>
</solver>



<!--#########################################################################
  AGENT GRID SECTION
#########################################################################-->

<!-- The agent grid contains and manages all agents living in the given domain.
  The parameters should be adjusted to match the simulation conditions. -->

<agentGrid>

  <param name="computationDomain">MyBiofilm</param>
  <param name="resolution" unit="um">4</param>

  <detachment class="DS_Quadratic">
    <param name="kDet" unit="um-1.hour-1">5e-6</param>
    <param name="maxTh" unit="um">200</param>
  </detachment>
  <param name="sloughDetachedBiomass">false</param>

  <!-- These parameters relate to particle shoving and are generally okay as-is. -->
  <param name="shovingMaxNodes">2e6</param>
  <param name="shovingFraction">0.025</param>
```

```
  <param name="shovingMaxIter">250</param>
  <param name="shovingMutual">true</param>
 </agentGrid>


 <!--#########################################################################
  SPECIES SECTION
 #########################################################################-->

<!--#########################################################################
  PLASMIDS
 #########################################################################-->

  <species class="Episome" name="MyPlasmid">
  <!--  iDynomics required fields   -->
  <param name="pilusLength">0.00000005</param>
  <param name="exchangeLag">0.333333333</param><!--10 sec 0.002777778-->
  <param name="receptionLag">0.002777778</param>
  <param name="lossProbability">0.000000001</param>
  <param name="transferProficiency">0.15</param>
  <param name="transProfChangeHour" unit="hour-1">2.10</param>
  <param name="transProfNew">0.80</param>
  <param name="nCopy">1</param>
  <param name="compatibilityMarker">1</param>

  <conjugationZone name="zone1">
   <param name="OriT">oritj</param>

   <param name="T4SS">T4SSi</param>
   <param name="precondition"></param>
  </conjugationZone>

  <gene name="G0">
   <param name="longName">gene 0</param>
   <promoter name="PromoterZone">
    <param name="promoterType">inhibitor</param>
    <param name="precondition"></param>
   </promoter>
   <param name="outputName">P0</param>
   <param name="outputTime" unit="minute">1</param>
   <param name="geneType">Protein</param>
  </gene>
 </species>

 <species class="Episome" name="MyPlasmid4">
  <!--  iDynomics required fields   -->
  <param name="pilusLength">0.0</param>
  <param name="exchangeLag">0.333333333</param>
  <param name="receptionLag">0.002777778</param>
  <param name="lossProbability">0.000000001</param>
  <param name="transferProficiency">0</param>
  <param name="nCopy">1</param>
  <param name="compatibilityMarker">1</param>
```

```
  <conjugationZone name="zone1">
   <param name="precondition"></param>
  </conjugationZone>
  <gene name="T4SSi">
   <param name="longName">T4SS1</param>
   <promoter name="PromoterZone">
    <param name="promoterType">inhibitor</param>
    <param name="precondition"></param>
   </promoter>
   <param name="outputName">T4SSi</param>
   <param name="outputTime" unit="minute">1</param>
   <param name="geneType">Protein</param>
  </gene>
 </species>


<!--#############################################################################
  BACTERIA
 #############################################################################-->


 <species class="EpiBac" name="MyAutotroph">
  <particle name="biomass">
   <param name="mass" unit="fg">1000</param>
  </particle>
  <particle name="inert">
   <param name="mass" unit="fg">100</param>
  </particle>

  <param name="donorColor">LightGrey</param>
  <param name="transconjugantColor">Aquamarine</param>
  <param name="recipientColor">LightGrey</param>

  <param name="computationDomain">MyBiofilm</param>
  <param name="divRadius" unit="um">2</param>
  <param name="deathRadius" unit="um">0.2</param>
  <param name="shoveFactor" unit="um">1.15</param>
  <param name="shoveLimit" unit="um">0.</param>
  <param name="scanSpeed" unit="um">1.0</param>

  <param name="proteinConcentrationMax">100.0</param>
  <param name="proteinConcentrationMin">0.1</param>
  <param name="penaltyGrowth">0</param>
  <param name="penaltyTime" unit="hour">0.0</param>
  <param name="minConjugationThreshold" unit="hour">0.416666667</param>
  <param name="maxConjugationThreshold" unit="hour">0.666666667</param>

  <plasmid name="MyPlasmid"></plasmid>
  <plasmid name="MyPlasmid4"></plasmid>

  <param name="epsMax">0.1</param>
  <param name="kHyd" unit="hr-1">0.007</param>
```

```
  <reaction name="MyGrowthAutotrophs" status="active"/>

  <initArea number="50">
   <param name="birthday" unit="hour">0</param>
   <coordinates x="0" y="0" z="0"/>
   <coordinates x="0" y="130" z="130"/>
  </initArea>
 </species>

 <species class="EpiBac" name="MyAutotroph2">
  <particle name="biomass">
   <param name="mass" unit="fg">1000</param>
  </particle>
  <particle name="inert">
   <param name="mass" unit="fg">100</param>
  </particle>

  <param name="donorColor">DimGray</param>
  <param name="transconjugantColor">green</param>
  <param name="recipientColor">DimGray</param>

  <param name="computationDomain">MyBiofilm</param>
  <param name="divRadius" unit="um">2</param>
  <param name="deathRadius" unit="um">0.2</param>
  <param name="shoveFactor" unit="um">1.15</param>
  <param name="shoveLimit" unit="um">0.</param>
  <param name="scanSpeed" unit="um">1.0</param>

  <param name="proteinConcentrationMax">100.0</param>
  <param name="proteinConcentrationMin">0.1</param>
  <param name="penaltyGrowth">0</param>
  <param name="penaltyTime" unit="hour">0.0</param>
  <param name="minConjugationThreshold" unit="hour">0.416666667</param>
  <param name="maxConjugationThreshold" unit="hour">0.666666667</param>

  <param name="epsMax">0.1</param>
  <param name="kHyd" unit="hr-1">0.007</param>
  <reaction name="MyGrowthAutotrophs" status="active"/>
  <initArea number="50">
   <param name="birthday" unit="hour">0</param>
   <coordinates x="0" y="0" z="0"/>
   <coordinates x="0" y="130" z="130"/>
  </initArea>
 </species>
</idynomics>
```

Sample AND Circuit Protocol File

```
<?xml version="1.0" encoding="UTF-8"?>
<!--

#########################################################################
 iDynoMiCS: individual-based Dynamics of Microbial Communities Simulator
#########################################################################

  Website: http://www.idynomics.org

-->


<!-- The entire simulation description is contained within the <idynomics> mark-up tags. -->
<idynomics>

 <!--#####################################################################
   SIMULATOR SECTION
 #####################################################################-->

 <simulator>

   <param name="restartPreviousRun">false</param>
   <!-- <param name="randomSeed">42</param> -->
   <param name="outputPeriod" unit="hour">0.416666667</param>

   <timeStep>
     <param name="adaptive">false</param>
     <param name="timeStepIni" unit="hour">0.416666667</param>
     <param name="timeStepMin" unit="hour">0.025</param>
     <param name="timeStepMax" unit="hour">1</param>
     <param name="endOfSimulation" unit="hour">6.25</param>
   </timeStep>
   <param name="agentTimeStep" unit="hour">0.044166667</param>
 </simulator>


 <!--#####################################################################
   INPUT SECTION
 #####################################################################-->


 <input>
   <param name="useAgentFile">false</param>
   <param name="inputAgentFileURL">agent_State(last).xml</param>
   <param name="useBulkFile">false</param>
   <param name="inputBulkFileURL">env_Sum(last).xml</param>
 </input>



 <!--#####################################################################
```

```
 SOLUTES AND BIOMASS TYPES SECTION
######################################################################-->


<solute domain="MyBiofilm" name="MyCOD">
  <param name="diffusivity" unit="m2.day-1">1e-4</param>
</solute>
<solute domain="MyBiofilm" name="pressure">
  <param name="diffusivity" unit="m2.day-1">0</param>
</solute>



<particle name="biomass">
  <param name="density" unit="g.L-1">150</param>
</particle>
<particle name="inert">
  <param name="density" unit="g.L-1">150</param>
</particle>



<!--######################################################################
  WORLD SECTION
######################################################################-->


<world>
  <bulk name="MyTank">

    <param name="isConstant">true</param>
    <param name="D" unit="h-1">0.6</param>

    <solute name="MyCOD">
      <param name="Sbulk" unit="g.L-1">10e-3</param>
      <param name="Sin" unit="g.L-1">10e-3</param>
    </solute>
    <solute name="pressure">
      <param name="Sbulk" unit="g.L-1">0</param>
      <param name="Sin" unit="g.L-1">0</param>
    </solute>
  </bulk>

  <!-- The computation domain is a physical region that will contain the biofilm,
    and has a more complex setup. -->

  <computationDomain name="MyBiofilm">

    <grid nDim="3" nI="33" nJ="33" nK="33"/>
    <param name="resolution" unit="um">4</param>
    <param name="boundaryLayer" unit="um">40</param>
    <param name="biofilmDiffusivity">0.8</param>
    <param name="specificArea" unit="m2.m-3">80</param>
```

```
   <boundaryCondition class="BoundaryZeroFlux" name="y0z">
    <shape class="Planar">
     <param name="pointIn" x="-1" y="0" z="0"/>
     <param name="vectorOut" x="-1" y="0" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryBulk" name="yNz">
    <param name="activeForSolute">yes</param>
    <param name="bulk">MyTank</param>
    <shape class="Planar">
     <param name="pointIn" x="33" y="0" z="0"/>
     <param name="vectorOut" x="1" y="0" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryCyclic" name="x0z">
    <shape class="Planar">
     <param name="pointIn" x="0" y="-1" z="0"/>
     <param name="vectorOut" x="0" y="-1" z="0"/>
    </shape>
    <shape class="Planar">
     <param name="pointIn" x="0" y="33" z="0"/>
     <param name="vectorOut" x="0" y="1" z="0"/>
    </shape>
   </boundaryCondition>

   <boundaryCondition class="BoundaryCyclic" name="x0y">
    <shape class="Planar">
     <param name="pointIn" x="0" y="0" z="-1"/>
     <param name="vectorOut" x="0" y="0" z="-1"/>
    </shape>
    <shape class="Planar">
     <param name="pointIn" x="0" y="0" z="33"/>
     <param name="vectorOut" x="0" y="0" z="1"/>
    </shape>
   </boundaryCondition>

  </computationDomain>
</world>



<!--#############################################################################
  REACTION SECTION
##############################################################################-->


<reaction catalyzedBy="biomass" class="ReactionFactor" name="MyGrowthAutotrophs">
  <param name="muMax" unit="hour">0.044166667</param>
  <kineticFactor class="MonodKinetic" solute="MyCOD">
    <param name="Ks" unit="g.L-1">2.5e-4</param>
```

```
 </kineticFactor>
 <yield>
  <param name="MyCOD" unit="g.g-1">-1.5</param>
  <param name="biomass" unit="g.g-1">1</param>
 </yield>
</reaction>



<!--#########################################################################
 SOLVER SECTION
#########################################################################-->
<solver class="Solver_multigrid" name="solutes" domain="MyBiofilm">
 <param name="active">true</param>
 <param name="preStep">150</param>
 <param name="postStep">150</param>
 <param name="coarseStep">1500</param>
 <param name="nCycles">5</param>

 <reaction name="MyGrowthAutotrophs"/>

</solver>

<solver class="Solver_pressure" name="pressure" domain="MyBiofilm">
 <param name="active">true</param>
</solver>



<!--#########################################################################
 AGENT GRID SECTION
#########################################################################-->

<!-- The agent grid contains and manages all agents living in the given domain.
 The parameters should be adjusted to match the simulation conditions. -->

<agentGrid>

 <param name="computationDomain">MyBiofilm</param>
 <param name="resolution" unit="um">4</param>

 <detachment class="DS_Quadratic">
  <param name="kDet" unit="um-1.hour-1">5e-6</param>
  <param name="maxTh" unit="um">200</param>
 </detachment>
 <param name="sloughDetachedBiomass">false</param>

 <!-- These parameters relate to particle shoving and are generally okay as-is. -->
 <param name="shovingMaxNodes">2e6</param>
 <param name="shovingFraction">0.025</param>
 <param name="shovingMaxIter">250</param>
 <param name="shovingMutual">true</param>
```

```
 </agentGrid>


 <!--#########################################################################
  SPECIES SECTION
 #########################################################################-->

<!--#########################################################################
  PLASMIDS
 #########################################################################-->

  <species class="Episome" name="ProteinX">
  <!--  iDynomics required fields   -->
  <param name="pilusLength">0.00000005</param>
  <param name="exchangeLag">0.333333333</param><!--10 sec 0.002777778-->
  <param name="receptionLag">0.002777778</param>
  <param name="lossProbability">0.000000001</param>
  <param name="transferProficiency">0.15</param>
  <param name="transProfChangeHour" unit="hour-1">2.10</param>
  <param name="transProfNew">0.80</param>
  <param name="nCopy">1</param>
  <param name="compatibilityMarker">1</param>

  <conjugationZone name="zone1">
   <param name="OriT">oritj</param>
   <param name="precondition"></param>
  </conjugationZone>

  <gene name="proteinX">
   <param name="longName">Protein X</param>
   <promoter name="PromoterZone">
    <param name="promoterType">inhibitor</param>
    <param name="precondition"></param>
   </promoter>
   <param name="outputName">pX</param>
   <param name="outputTime" unit="minute">1</param>
   <param name="geneType">Protein</param>
  </gene>
 </species>

 <species class="Episome" name="ProteinY">
  <!--  iDynomics required fields   -->
  <param name="pilusLength">0.00000005</param>
  <param name="exchangeLag">0.333333333</param><!--10 sec 0.002777778-->
  <param name="receptionLag">0.002777778</param>
  <param name="lossProbability">0.000000001</param>
  <param name="transferProficiency">0.15</param>
  <param name="transProfChangeHour" unit="hour-1">2.10</param>
  <param name="transProfNew">0.80</param>
  <param name="nCopy">1</param>
  <param name="compatibilityMarker">2</param>

  <conjugationZone name="zone1">
```

```
    <param name="OriT">oritk</param>
    <param name="precondition"></param>
   </conjugationZone>

   <gene name="proteinY">
    <param name="longName">Protein Y</param>
    <promoter name="PromoterZone">
     <param name="promoterType">inhibitor</param>
     <param name="precondition"></param>
    </promoter>
    <param name="outputName">pY</param>
    <param name="outputTime" unit="minute">1</param>
    <param name="geneType">Protein</param>
   </gene>
  </species>

  <species class="Episome" name="ReporterAND">
   <!--  iDynomics required fields   -->
   <param name="pilusLength">0.00000005</param>
   <param name="exchangeLag">0.333333333</param><!--10 sec 0.002777778-->
   <param name="receptionLag">0.002777778</param>
   <param name="lossProbability">0.000000001</param>
   <param name="transferProficiency">0.15</param>
   <param name="transProfChangeHour" unit="hour-1">2.10</param>
   <param name="transProfNew">0.80</param>
   <param name="nCopy">1</param>
   <param name="compatibilityMarker">3</param>

   <conjugationZone name="zone1">
    <param name="OriT">oritL</param>
    <param name="precondition">pX AND pY</param>
   </conjugationZone>

   <gene name="rAND">
    <param name="longName">Reporter AND</param>
    <promoter name="PromoterZone">
     <param name="promoterType">activator</param>
     <param name="precondition">pX AND pY</param>
    </promoter>
    <param name="outputName">rAND</param>
    <param name="outputTime" unit="minute">1</param>
    <param name="geneType">Reporter</param>
    <param name="color">green</param>
   </gene>
  </species>



<!--#############################################################################
   BACTERIA
  ##############################################################################-->
```

```
<species class="EpiBac" name="MyAutotrophX">
 <particle name="biomass">
  <param name="mass" unit="fg">1000</param>
 </particle>
 <particle name="inert">
  <param name="mass" unit="fg">100</param>
 </particle>

 <param name="donorColor">LightGrey</param>
 <param name="transconjugantColor">LightGrey</param>
 <param name="recipientColor">LightGrey</param>

 <param name="computationDomain">MyBiofilm</param>
 <param name="divRadius" unit="um">2</param>
 <param name="deathRadius" unit="um">0.2</param>
 <param name="shoveFactor" unit="um">1.15</param>
 <param name="shoveLimit" unit="um">0.</param>
 <param name="scanSpeed" unit="um">1.0</param>

 <param name="proteinConcentrationMax">100.0</param>
 <param name="proteinConcentrationMin">0.1</param>
 <param name="penaltyGrowth">0</param>
 <param name="penaltyTime" unit="hour">0.0</param>
 <param name="minConjugationThreshold" unit="hour">0.333333333</param>
 <param name="maxConjugationThreshold" unit="hour">0.666666667</param>

 <plasmid name="ProteinX"></plasmid>

 <param name="epsMax">0.1</param>
 <param name="kHyd" unit="hr-1">0.007</param>

 <reaction name="MyGrowthAutotrophs" status="active"/>

 <initArea number="12">
  <param name="birthday" unit="hour">0</param>
  <coordinates x="0" y="0" z="0"/>
  <coordinates x="0" y="130" z="130"/>
 </initArea>
</species>

<species class="EpiBac" name="MyAutotrophY">
 <particle name="biomass">
  <param name="mass" unit="fg">1000</param>
 </particle>
 <particle name="inert">
  <param name="mass" unit="fg">100</param>
 </particle>

 <param name="donorColor">DimGray</param>
 <param name="transconjugantColor">DimGray</param>
 <param name="recipientColor">DimGray</param>

 <param name="computationDomain">MyBiofilm</param>
```

```
  <param name="divRadius" unit="um">2</param>
  <param name="deathRadius" unit="um">0.2</param>
  <param name="shoveFactor" unit="um">1.15</param>
  <param name="shoveLimit" unit="um">0.</param>
  <param name="scanSpeed" unit="um">1.0</param>

  <param name="proteinConcentrationMax">100.0</param>
  <param name="proteinConcentrationMin">0.1</param>
  <param name="penaltyGrowth">0</param>
  <param name="penaltyTime" unit="hour">0.0</param>
  <param name="minConjugationThreshold" unit="hour">0.333333333</param>
  <param name="maxConjugationThreshold" unit="hour">0.666666667</param>

  <param name="epsMax">0.1</param>
  <param name="kHyd" unit="hr-1">0.007</param>

  <plasmid name="ProteinY"></plasmid>

  <reaction name="MyGrowthAutotrophs" status="active"/>

  <initArea number="12">
   <param name="birthday" unit="hour">0</param>
   <coordinates x="0" y="0" z="0"/>
   <coordinates x="0" y="130" z="130"/>
  </initArea>
 </species>

 <species class="EpiBac" name="MyAutotrophAND">
  <particle name="biomass">
   <param name="mass" unit="fg">1000</param>
  </particle>
  <particle name="inert">
   <param name="mass" unit="fg">100</param>
  </particle>

  <param name="donorColor">Gray</param>
  <param name="transconjugantColor">Gray</param>
  <param name="recipientColor">Gray</param>

  <param name="computationDomain">MyBiofilm</param>
  <param name="divRadius" unit="um">2</param>
  <param name="deathRadius" unit="um">0.2</param>
  <param name="shoveFactor" unit="um">1.15</param>
  <param name="shoveLimit" unit="um">0.0</param>
  <param name="scanSpeed" unit="um">1.0</param>

  <param name="proteinConcentrationMax">100.0</param>
  <param name="proteinConcentrationMin">0.1</param>
  <param name="penaltyGrowth">0</param>
  <param name="penaltyTime" unit="hour">0.0</param>
  <param name="minConjugationThreshold" unit="hour">0.333333333</param>
  <param name="maxConjugationThreshold" unit="hour">0.666666667</param>
```

```
   <plasmid name="ReporterAND"></plasmid>

   <param name="epsMax">0.1</param>
   <param name="kHyd" unit="hr-1">0.007</param>

   <reaction name="MyGrowthAutotrophs" status="active"/>

   <initArea number="12">
    <param name="birthday" unit="hour">0</param>
    <coordinates x="0" y="0" z="0"/>
    <coordinates x="0" y="130" z="130"/>
   </initArea>
  </species>

  <species class="EpiBac" name="MyAutotroph">
   <particle name="biomass">
    <param name="mass" unit="fg">1000</param>
   </particle>
   <particle name="inert">
    <param name="mass" unit="fg">100</param>
   </particle>

   <param name="donorColor">Gray35</param>
   <param name="transconjugantColor">Gray35</param>
   <param name="recipientColor">Gray35</param>

   <param name="computationDomain">MyBiofilm</param>
   <param name="divRadius" unit="um">2</param>
   <param name="deathRadius" unit="um">0.2</param>
   <param name="shoveFactor" unit="um">1.15</param>
   <param name="shoveLimit" unit="um">0.0</param>
   <param name="scanSpeed" unit="um">1.0</param>

   <param name="proteinConcentrationMax">100.0</param>
   <param name="proteinConcentrationMin">0.1</param>
   <param name="penaltyGrowth">0</param>
   <param name="penaltyTime" unit="hour">0.0</param>
   <param name="minConjugationThreshold" unit="hour">0.333333333</param>
   <param name="maxConjugationThreshold" unit="hour">0.666666667</param>

   <param name="epsMax">0.1</param>
   <param name="kHyd" unit="hr-1">0.007</param>

   <reaction name="MyGrowthAutotrophs" status="active"/>

   <initArea number="324">
    <param name="birthday" unit="hour">0</param>
    <coordinates x="0" y="0" z="0"/>
    <coordinates x="0" y="130" z="130"/>
   </initArea>
  </species>
 </idynomics>
```